U.S. Department of Interior
Geological Survey Open-File Report 80-228

GARNET--Computer Applications Software for the National Coal Resources Data System

by A. C. Olson

GARNET (Graphic Analysis of Resources with Numerical Evaluation Techniques) is a set of interactive graphics programs that have been written at the U.S. Geological Survey during the past four years. This software system is designed to aid the commodity geologist in analyzing and evaluating resources when dealing with irregularly spaced, point-located field data.

After the coordinates of the field observations and related measurements have been entered into a data file, the geologist can generate an interpolated grid surface based on measured values at each of the observation points. From these interpolated surfaces, the geologist can produce maps of the coal-bed structure elevations, coal-bed thickness (isopach), ratio of thickness of overburden to bed thickness, and chemical concentrations as well as the ultimate resource map.

Outcrop and political boundaries can be added to the data set by means of a digitizer. Thickness, overburden, or chemical concentration boundaries can be created from the interpolated grid data. Boundaries can also be created by means of the cursor during a session at the graphics terminal. The software permits all logical combinations of bounded areas to form additional boundaries for resource delineation.

Additional features include the capability of generating trend surfaces to aid in analyzing data tendencies, and an editing capability to permit changes to the observed data.

GARNET was designed to meet the growing need for more accurate and more rapid computation of coal resource inventories. It automates those time-consuming tasks that heretofore the geologist performed manually. The calculations and subsequent map displays can be accomplished during a single session at an interactive graphics terminal. An option is provided for creating a plot tape to produce maps on an off-line plotter. Volume and tonnage values can be computed for each reliability category and for each boundary set.

The author is most deeply indebted to R. J. Smith who has provided invaluable programming support and assistance for GARNET during the past year. In the future, Mr. Smith will be responsible for maintaining the system.

aco

1/24/80
Reston, Va.

```
dimension           alfa(8),        c(69),          radius(10)
dimension           iarray(20)
double precision    vol(10),        volm
double precision    optn(12),       comand
common /switch/      isobs,icrt,iwind,itek,i4027,
                    is6,is7,is8,is9,is10
common /surfac/     xob(1500),      yob(1500),
     zt(1500),      zob(1500),      residu(1500),
     m,             n,              np,
     perd,          resum,          ntc,
     npc,           msq,            msq2,
     msq3,          msq4,           sig
common /graphc/     xmin,           ymin,           zmin,
     zgrid(65000),  xmax,           ymax,           zmax,
     istat(65000),  nx,             ny,             nz,
     x(16000),      nxgd,           nygd,           ncell,
     y(16000),      delgrd,         igrid,          icell,
     into,          zint,           zlevel,         level,
     nbold,         lplot,          dashl,          pcrang,
     levmin,        levmax,         ifit,           kset,
     ndig,          ktour,          ismth,          icnt
common /corner/     xcorn(4),       ycorn(4)
common /factor/     scale
common /baud/       ibaud
common /faultl/     xf(200),        yf(200),        zft(200),
                    nfp(10)
common /bound/ xb(20000,2),yb(20000,2),nch(2000,2),noch(2),
```

```
      kod(2000,2),ipar(2),icnd(2),kap(15,2),nptot(2)
      common /files/ filnm1,filnm2,filnm3,filgrd,filobs
      character*8 filnm1,filnm2,filnm3,filgrd,filobs,file
      data optn/"trend","edit","grid","contour","gridop","resource",
     1     "volume","location","boundary","window","unwind","quit"/
c     call acct("garnet")
50    call prompt("ENTER TRANSMISSION RATE (BAUD): ",32)
      read (5,1018,err=50) ibaud
55    call prompt("ENTER '1' IF USING A TEKTRONIX 4010, '2' FOR 4014/4015: ",56)
      read(5,1010,err=55) itest
      itek      =  iabs(itest)
      i4027     =  (1 - isign(1,itest))/2
      call initt(ibaud/10)
      iwind     =  0
      if(lplot .le. 1)  call anmode
      call chrsiz(4)
      write(6,1020)
60    call prompt("TO CONTINUE, PRESS RETURN KEY: ",31)
      read(5,1005,err=60)  itest
      if(itest .eq. 23) go to 65
      call newpag
      if(lplot .le. 1)  call anmode
      call assoc(33,"noteg1  ","si  ")
61    read(33,1001,end=62)  iarray
      write(6,1002)  iarray
      go to 61
62    call closer(33)
65    call prompt("FOR COMMAND LIST, PRESS RETURN KEY: ",36)
      read(5,1005,err=65)  itest
      if(itest .eq. 23) go to 100
      call newpag
      if(lplot .le. 1)  call anmode
      call assoc(33,"noteg2  ","si  ")
66    read(33,1001,end=67)  iarray
      write(6,1002)  iarray
      go to 66
67    call closer(33)
70    call prompt("WHEN READY TO PROCEED, ENTER CARRIAGE RETURN.",45)
      read(5,1008,err=70)  itest
100   call newpag
      if(lplot .le. 1)  call anmode
      if(i4027 .eq. 1)  write(6,40271)
      call prompt("ENTER COMMAND: ",15)
      read(5,1019)comand
      dashl     =  .05
      isobs     =  0
      icrt      =  0
      lplot     =  0
      ismth     =  0
      ktour     =  0
      imap      =  0
      do 200 i=1,12
      kk        =  i
200   if(comand .eq. optn(i)) go to 300
      go to 100
300   go to (2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,3100,9999),kk
2100  write(6,1020)
      call obsin(c)
      call surfit(c)
      call newpag
```

```
      if(lplot .le. 1)  call anmode
      write(6,1006) sig
      call messag
      write(6,1007)
2110  call prompt("(yes or no): ",13)
      read(5,1008)itest
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2110
      if(itest .eq. "no") go to 100
2115  write(6,2101)
2101  format(" ENTER FRACTION OF MAP TO BE PLOTTED OUTSIDE OF"
      " RECTANGULAR BOUNDARY: ")
      read(5,1009,err=2115) pcrang
      go to 3200
2120  write(6,1020)
      call prompt("GRID SIZE: ",11)
      read (5,1010,err=2120)delgrd
      call gridit(c)
      do 2130 i=1,nz
2130  if(zgrid(i) .lt. -.1e-10) zgrid(i)  =  -.1e-10
      if(zmin .lt. 0.) zmin  =  0.
      write(6,1020)
      call obsin$obsout(c)
      write(6,1020)
      call gridin$grido
      go to 100
2200  write(6,2201)
2201  format(" HAS A TREND SURFACE BEEN DONE FOR THESE POINTS?")
2210  call prompt("(yes or no): ",13)
      read(5,1008)itest
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2210
      if(itest .eq. "yes") go to 2400
      write(6,2202)
2202  format(" A TREND SURFACE FIT IS NEEDED TO FLAG THE MOST"
      " DEVIANT POINT VALUES."/" ENTER ""trend"" WHEN PROMPTED FOR"
      " COMMAND.  AFTER THE TREND SURFACE"/" IS OBTAINED, THE"
      " EDITING PROCEDURE MAY BE USED.")
      go to,100
2250  kset    =  1
      kap(15,1)=  "no"
      if(lplot .le. 1)  call anmode
      if(i4027 .eq. 1)  write(6,40282)
      if(lplot .le. 1)  call anmode
      call ctour
      call editor
      call ndcopy
      call newpag
      if(lplot .le. 1)  call anmode
      write(6,2205)
2205  format(" DO YOU WANT TO CREATE A FILE FOR THE EDITED"
      " OBSERVATION DATA?")
2260  call prompt("(yes or no): ",13)
      read(5,1008)itest
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2260
      if(itest .eq. "yes") call obsin$obsout(c)
      go to 100
2300  write(6,2301)
2301  format(" DO YOU WISH TO WEIGHT THE NEW SET OF GRID VALUES WITH"
      " THE VALUES"/" FROM AN OLD GRID?" )
2310  call prompt("(yes or no): ",13)
      read(5,1008)itest
```

```
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2310
      if(itest .eq. "no") go to 2330
      write(6,2302)
2302  format(" HAS THE SET OF OLD GRID VALUES BEEN LOADED INTO THE"
     " GRID ARRAY?")
2320  call prompt("(yes or no): ",13)
      read(5,1008)itest
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2320
      if(itest .eq. "no") call gridin
      gfact   =  .0048
      go to 2340
2330  gfact   =  0.
2340  call obsin(c)
2355  write(6,2101)
      read(5,1009) pcrang
      go to 3200
2360  call prompt("GRID SIZE: ",11)
      read(5,1010,err=2360)  delgrd
      xnp     =  np
      nx      =  (xmax - xmin)/delgrd + 1.9999
      ny      =  (ymax - ymin)/delgrd + 1.9999
      nz      =  nx*ny
      gfact   =  gfact*sqrt(180./xnp)
      write(6,1020)
      write(6,2304)
2304  format(" DO YOU WISH TO USE THE DEFAULT GRIDDING METHOD?")
2365  call prompt("(yes or no): ",13)
      read(5,1008) itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 2365
      if(itest .eq. "no")  go to 2369
      call zntrpl
      go to 2395
2369  write(6,1020)
      write(6,2305)
2305  format(" IS THE DATA DENSE AND DISTRIBUTED UNIFORMLY?")
2370  call prompt("(yes or no): ",13)
      read(5,1008)  itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 2370
      if(itest .eq. "no")  go to 2375
      call qdgrid(gfact)
      go to 2395
2375  write(6,1020)
      write(6,2306)
2306  format(" DO YOU HAVE AT LEAST SEVEN OBSERVATION POINTS?")
2380  call prompt("(yes or no): ",13)
      read(5,1008)  itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 2380
      if(itest .eq. "yes")  go to 2385
      write(6,1020)
      write(6,2307)
2307  format(" YOU DO NOT HAVE ENOUGH POINTS TO DO A GRIDDED"
            " SURFACE. "/" WHEN READY TO CONTINUE, PRESS"
            " THE CARRIAGE RETURN.")
      go to 100
2385  write(6,1020)
      call planar(gfact)
2395  if(zmin .lt. 0.)  zmin = 0.
      zmn       =  zob(1)
      zmx       =  zob(1)
      do 2396 i=2,np
```

```
          if(zmn .gt. zob(i))  zmn = zob(i)
2396      if(zmx .lt. zob(i))  zmx = zob(i)
          do 2398 i=1,nz
          if(zmn .le. 0.)  go to 2397
          if(zgrid(i) .lt. zmn)  zgrid(i) = zmn*(1.-.25*(zgrid(i)-zmn)/(zmin-zmn))
2397      if(zgrid(i) .lt. 0.)  zgrid(i) = 0.
2398      if(zgrid(i) .gt. zmx)  zgrid(i) = zmx*(1.+.25*(zgrid(i)-zmx)/(zmax-zmx))
          zmax      =  1.25*zmx
          write(6,1020)
          write(6,2303)
2303      format(" A NEW GRIDDED DATA FILE HAS BEEN CREATED.  ENTER")
          write(6,1020)
          call gridin$gridc
          write(6,1020)
          go to 100
2400      if(kk .eq. 6)  go to 3300
          write(6,1020)
          call obsin(c)
2422      write(6,1020)
c         write(6,2423)
c 2423    format(" HAS GRIDDED DATA BEEN LOADED INTO GRID ARRAY?")
c 2424      call prompt("(yes or no): ",13)
c           read(5,1008)  itest
c         if(itest .ne. "yes" .and. itest .ne. "no") go to 2424
c           if(itest .eq. "no")  call gridin
          call gridin
          if(iwind .eq. 1)  call window$windsv
          if(iwind .eq. 0)  call window$unwind
2425      write(6,1020)
          call prompt("CONTOUR INTERVAL: ",18)
          read(5,1010,err=2425)  zint
2426      write(6,1020)
          call prompt("FREQUENCY OF BOLD LINES: ",25)
          read(5,1010,err=2426)  nbold
          scale  =  1.
2427      write(6,1020)
          call prompt("ENTER PLOT SIZE MULTIPLIER: ",28)
          read(5,1021,err=2427)  scale
          ktour     =  0
          if(scale .lt. 0.)  ktour = 1
          scale     =  abs(scale)
          if(scale .le. 0.) scale  =  1.
2428      write(6,1020)
          call prompt("FOR CONTOUR SMOOTHING, ENTER A ""1"": ",36)
          read(5,1015,err=2428)ismth
2429      write(6,1020)
c          call prompt("TO WRITE CONTOUR COORDINATES ONTO DISK, ENTER A ""1"": ",53)
c          read(5,1015,err=2429)ktour
c          write(6,1020)
          if(ktour .eq. 1)  call conin$coopen
          go to 3300
2430      if(i4027 .eq. 1)  write(6,40285)
          if(isobs .eq. 1 .and. imap .ne. 1)  call points
          kset    =  1
 call stty("-modes","^erkl,^ctl_char")
          if(i4027 .eq. 1)  write(6,40282)
          call ctour
 call stty("-modes","erkl,^ctl_char")
          if(imap.ne.1.and.kap(15,1).ne."no") is=2
          if(iplot .le. 1)  call anmode
```

```
      if(i4027 .eq. 1)  write(6,40284)
      if(lplot .le. 1)  call anmode
      if(imap .ne. 1.and.kap(15,1).ne."no")  call pltchn(is)
      if(ktour .eq. 1) call conin$eoclos
2440  go to (100,3200,100,2450,100,2650,100,100,100,100,100,9999),kk
2450  if(lplot .ge. 1) call calbeg$calend
    if(lplot.gt.1) go to 100
      if(lplot .le. 1) call hdcopy
      read(5,1005) idum
      call newpag
      if(lplot .le. 1)  call anmode
      call initt(ibaud/10)
      if(lplot .le. 1)  call anmode
      go to 100
2500  call binary($100)
      write(6,2501)
2501  format(1x,"A NEW GRIDDED DATA FILE HAS BEEN CREATED ..."/
           " DO YOU WISH TO SAVE IT?")
2510  call prompt("(yes or no): ",13)
      read(5,1008)itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 2510
      if(itest .eq. "yes")  call gridin$grido
      go to 100
2600  write(6,2601)
2601  format(" AT EACH PROMPT, ENTER THE RELIABILITY RADIUS IN"
      " DIGITIZER INCHES."/" ENTER THE SMALLEST RADIUS AT THE"
      " PROMPT AND CONTINUE TO ENTER THE RADII"/" IN ASCENDING"
      " ORDER.  AFTER THE LAST RADIUS HAS BEEN ENTERED, ENTER"/
      " AN ""0"" AT THE NEXT PROMPT TO TERMINATE THE INPUT"
      " PROCEDURE."//)
      do 2610 i=1,10
2606  write(6,2602) i
2602  format(" ENTER RADIUS FOR RELIABILITY CATEGORY",i2,".")
      read(5,1010,err=2606) radius(i)
      if(radius(i) .le. 0.) go to 2620
2610  krad    =  i
2620  write(6,1020)
      call obsin(c)
      write(6,1020)
      call gridin
      if(iwind .eq. 1)  call window$windsw
      if(iwind .eq. 0)  call window$unwind
      write(6,1020)
      write(6,2603)
2603  format(" DO YOU WISH TO PLOT A RESOURCE MAP?")
2635  call prompt("(yes or no): ",13)
      read(5,1008)itest
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2635
      if(itest .eq. "no") go to 3310
      imap    =  1
      write(6,1020)
      write(6,2605)
2605  format(" DO YOU WISH TO PLOT THE RESOURCE MAP WITH A"
      " CONTOUR SURFACE?")
2640  call prompt("(yes or no): ",13)
      read(5,1008)  itest
      if(itest .ne. "yes" .and. itest .ne. "no") go to 2640
      ictour  =  1
      if(itest .eq. "yes") go to 2425
      ictour  =  0
```

```
      go to 3300
2650  if(lplot .ge. 1) call newpen(3)
      do 2660 i=1,krad
      if(lplot .le. 1) call czaxis(1)
      npts    =' 122.*radius(i)
      if(lplot .ge. 1)  npts = 120
      if(lplot .le. 1)  call anmode
      if(i4027 .eq. 1)  write(6,40281)
      if(lplot .le. 1)  call anmode
      if(imap .eq. 1) call outlin(radius(i),npts)
      if(lplot .le. 1)  call anmode
      if(lplot .le. 1) call anmode
      call volume(radius(i),vol(i))
2660  vol(i)  =  vol(i)*4.e6
      if(scale .le. 1.)  scale = 1.
      if(i4027 .eq. 1)  write(6,40285)
      if(imap .eq. 1) call points
      if(imap.eq.1.and.kap(15,1).ne."no") is=2
      if(lplot .le. 1) call czaxis(1)
      if(lplot .ge. 1) call newpen(2)
      if(lplot .le. 1)  call anmode
      if(i4027 .eq. 1)  write(6,40284)
      if(lplot .le. 1)  call anmode
      if(imap .eq. 1.and.kap(15,1).ne."no")  call pltchn(is)
    if (lplot.gt.1) go to 2663
      call hdcopy
      read(5,1005) idum
      call newpag
      if(lplot .le. 1)  call anmode
2663  call prompt("ENTER TITLE (LESS THAN 32 CH)= ",31)
      read(5,1011)alfa
2665  call prompt("ENTER DENSITY TO CONVERT VOLUME TO TONNAGE: ",43)
      read(5,1010,err=2665)rho
    if(lplot.gt.1) go to 2667
      call newpag
      if(lplot .le. 1)  call anmode
2667  write(6,1012) alfa
      write(6,1013)
      r1      =  0.
      volm    =  0.d0
      do 2670 i=1,krad
      volm    =  vol(i) - volm
      tons    =  round(rho*volm)
      r2      =  radius(i)
      write(6,1014) i,r1,r2,volm,tons
      volm    =  vol(i)
2670  r1      =  r2
      go to 2450
2700  continue
2800  write(6,1020)
      call obsin(c)
      zmin    =  zob(1)
      zmax    =  zob(1)
      do 2820 i=2,np
      if(zob(i) .gt. zmax)  zmax = zob(i)
      if(zob(i) .lt. zmin)  zmin = zob(i)
2820  continue
2851  write(6,1020)
      call prompt("ENTER PLOT SIZE MULTIPLIER: ",28)
      read(5,1021,err=2851)  scale
```

```
      if(scale .le. 0.)  scale = 1.
2852  write(6,1020)
      write(6,2101)
      read(5,1009,err=2852)  perang
      go to 3300
2850  if(i4027 .eq. 1)  write(6,40285)
      call points
      call hdcopy
      read(5,1005)  idum
      go to 100
2900  kset    =  -1
      call boundary
      go to 100
3000  call window
      go to 100
3100  call window$unwind
      go to 100
3300  write(6,1020)
      write(6,1017)
      call prompt("GRAPHICS DEVICE? (0,1 OR 2): ",29)
      read(5,1010)lplot
3200      if(kk .eq. 2) go to 3325
      if(kk .ge. 8)  go to 3324
      if(kk .ne. 4  .and.  kk .ne. 6) go to 3330
3310  write(6,1020)
      is=2
      call chnin(is,ierr)
      if(ierr.eq.0) go to 3324
      call slice(xmin,xmax,is)
3324  if(lplot .ge. 1) call calbeg
3325  if(i4027 .eq. 0)  go to 3326
      write(6,40270)
      write(6,40272)
      write(6,40273)
      write(6,40283)
3326  if(lplot .le. 1) call crtbeg(xmin,xmax,ymin,ymax)
      ndig    =  abs(aint(alog10(zmax) + 1.))
      if(lplot .le. 1)  call anmode
      if(ictour .eq. 1 .and. kk .eq. 6) go to 2430
      if(lplot .le. 1)  call anmode
3330  go to (2120,2250,2360,2430,100,2650,100,2850,100,100,100,9999),kk
9999  return
40270 format(" !WOR 33 H")
40271 format(" !MON 34 H K")
40272 format(" !GRA 1,35")
40273 format(" !SHR")
40274 format(" !DEL 100")
40280 format(" !COL C0")
40281 format(" !COL C1")
40282 format(" !COL C2")
40283 format(" !COL C3")
40284 format(" !COL C4")
40285 format(" !COL C5")
40286 format(" !COL C6")
40287 format(" !COL C7")
1001  format(20a4)
1002  format(5x,20a4)
1005  format(i2)
1006  format(/" THE SURFACE HAS BEEN FITTED TO THE DATA.  THE ROOT"/
      " MEAN SQUARE ERROR IS ",f7.3," ."/)
```

```
1007      format(" DO YOU WISH TO HAVE THIS TREND SURFACE GRIDDED?")
1008      format(a4)
1009      format(f10.2)
1010      format(v)
1011      format(8a4)
1012      format(10x,"TABLE OF RESOURCE ESTIMATES FOR ",8a4//)
1013      format(23x,"RADIUS",12x,"VOLUME",14x,"TONNAGE"/21x,
          "-----------",5x,"---------------",5x,"---------------"//)
1014      format(5x,"CATEGORY NO. ",i1,2x,f5.2,"-",f5.2,5x,e15.6,5x,e15.6)
1015      format(i1)
1016      format(1x,"TO READ THE COMPACTED OBSERVED DATA FILE, PRESS"
               " THE CARRIAGE RETURN.")
1017      format(1x,"TO SPECIFY GRAPHICS DEVICE, "
                  "ENTER ""0"" FOR TEKTRONIX ONLY,"/29x,
                  "ENTER ""1"" FOR TEKTRONIX AND CALCOMP,"/26x,
               "OR ENTER ""2"" FOR CALCOMP ONLY.")
1018      format(i4)
1019      format(a8)
1020      format(/)
1021      format(f10.9)
          end
```

```
      subroutine adjust(um,vm,u,v)
c
c           This subroutine was written and programmed by
c                              A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
c
c  THIS SUBROUTINE ESTIMATES THE SLOPE AT THE END POINTS OF A
c  NON-CLOSED CURVE.  IT ASSUMES A CONSTANT RADIUS OF CURVATURE
c  THROUGH THE FIRST THREE OR THROUGH THE LAST THREE POINTS OF
c  THE CURVE.  GIVEN THE DIFFERENCE VECTOR, (UM,VM), FOR THE
c  ENDMOST PAIR OF POINTS, AND THE DIFFERENCE VECTOR, (U,V), FOR
c  THE SECOND AND THIRD POINTS FROM THE END, THIS SUBROUTINE
c  COMPUTES AND RETURNS A PSEUDO-DIFFERENCE VECTOR, ALSO (U,V),
c  FOR THE ENDPOINT AND A PHANTOM POINT BEYOND THE ENDPOINT.
c  THE NEW DIFFERENCE VECTOR IS EFFECTIVELY THE REFLECTION OF THE
c  OLD DIFFERENCE VECTOR, (U,V), ABOUT AN AXIS DEFINED BY THE
c  VECTOR, (UM,VM).
c
      dum1  =  2.0*um*vm
      dum2  =  um**2 - vm**2
      dum3  =  1.0/(um**2 + vm**2)
      q     =  (dum1*v + dum2*u)*dum3
      v     =  (dum1*u - dum2*v)*dum3
      u     =  q
      return
      end
```

```
      subroutine assoc(iunit,name,mode)
c
c          This subroutine was programmed by
c                    Roger Bowen
c          for general interactive applications of the MULTICS
c          system.
c
c
c iunit = fortran i/o unit number
c name = name of disk file data set. may be passed as a character
c        string literal with 8 characters or as a double precision
c        variable or as a character string variable.
c mode = "si   " for formatted input
c        "so   " for formatted output
c        "sqi  " for unformatted input
c        "sqo  " for unformatted output
c        "di   " for keyed input
c        "do   " for keyed output
c
      character*8 name,fname*6,mode*4,fmt*12
      fmt="(4hfile,i2) "
      if(iunit.le.9) fmt="(5hfile0,i1)"
      encode(fname,fmt) iunit
      call io_call("attach",fname,"vfile_ ",name)
      call io_call("open",fname,mode)
      go to 1
      entry closer(iunit)
      endfile iunit
      fmt="(4hfile,i2)"
      if(iunit.le.9) fmt="(5hfile0,i1)"
      encode(fname,fmt) iunit
      call io_call("close",fname)
      call io_call("detach",fname)
1     return
      end
```

```
      subroutine binary(*)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         iplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          isw1,
     8     isw2,          isw3,          isw4,          isw5
      common /param/      nx1,      ny1,      nz1,
     1                    nx2,      ny2,      nz2,
     2     xmin1,   xmax1,   ymin1,   ymax1,   zmin1,   zmax1,
     3     xmin2,   xmax2,   ymin2,   ymax2,   zmin2,   zmax2,
     4     delgd1,  xcorn1(4),        ycorn1(4),
     5     delgd2,  xcorn2(4),        ycorn2(4)
      dimension    z1(5),      z2(5)
      double precision filnam
      zmin    =  1.e20
      zmax    = -1.e20
1004  format(3i5/8f10.4/8f10.4)
1005  format(5e15.7)
300   write(6,1002)
1002  format(/1x,"ENTER THE OPERATION SYMBOL FOR ""SURFACE ONE"" "
     1     " (+,-,*,/) ""SURFACE TWO"":" )
      read(5,1001) iopn
      if(iopn .eq. " ") return 1
1001  format(a1)
      rewind 21
      rewind 22
      write(6,1003)
1003  format(1x,"ENTER FILE NAME FOR")
100   call prompt("SURFACE ONE: ",13)
      read(5,1008)filnam
1008  format(a8)
      call assoc(21,filnam,"si  ")
      read(21,1004,err=700) nx1,ny1,nz1,xmin1,xmax1,ymin1,ymax1,zmin1,zmax1,
     1     delgd1,pcrang,xcorn1,ycorn1
150   call prompt("SURFACE TWO: ",13)
      read(5,1008)filnam
      call assoc(22,filnam,"si  ")
      read(22,1004,err=800) nx2,ny2,nz2,xmin2,xmax2,ymin2,ymax2,zmin2,zmax2,
     1     delgd2,pcrang,xcorn2,ycorn2
```

```
      ineg    =  0
      write(6,1006)
1006  format(/1x,"ARE NEGATIVE Z-VALUES ACCEPTABLE?")
200   call prompt("(yes or no): ",13)
      read(5,1007)itest
1007  format(a4)
      if(itest .ne. "yes"  .and.  itest .ne. "no") go to 200
      if(itest .eq. "yes") ineg = 1
      call regstr(itest,$650)
      xmin    =  xmin1
      xmax    =  xmax1
      ymin    =  ymin1
      ymax    =  ymax1
      delgrd  =  delgd1
      nx      =  nx1
      ny      =  ny1
      nz      =  nz1
      if(iopn .eq. "+") go to 410
      if(iopn .eq. "-") go to 430
      if(iopn .eq. "*") go to 450
      if(iopn .eq. "/") go to 470
      write(6,1401)
1401  format(/" BINARY OPERATION NOT PROPERLY SPECIFIED."/
     "  ENTER ONLY A ""+"", ""-"", ""*"", OR A ""/"".")
      go to 300
410   do 415 i=1,nz,5
      read(21,1005) z1
      read(22,1005) z2
      do 415 j=1,5
      indx    =  i + j - 1
415   zgrid(indx)  =  z1(j) + z2(j)
      go to 500
430   do 435 i=1,nz,5
      read(21,1005) z1
      read(22,1005) z2
      do 435 j=1,5
      indx    =  i + j - 1
435   zgrid(indx)  =  z1(j) - z2(j)
      go to 500
450   do 455 i=1,nz,5
      read(21,1005) z1
      read(22,1005) z2
      do 455 j=1,5
      indx    =  i + j - 1
455   zgrid(indx)  =  z1(j)*z2(j)
      go to 500
470   do 475 i=1,nz,5
      read(21,1005) z1
      read(22,1005) z2
      do 475 j=1,5
      indx    =  i + j - 1
      if(abs(z2(j)) .gt. 0.) go to 474
      zgrid(indx)  =  0.
      go to 475
474   zgrid(indx)  =  z1(j)/z2(j)
475   continue
500   do 600 i=1,nz
      if(zgrid(i) .lt. zmin) zmin = zgrid(i)
600   if(zgrid(i) .gt. zmax) zmax = zgrid(i)
      if(ineg .ne. 1  .and.  zmin .lt. 0.) zmin = -1.e-10
```

```
650       call assoc$closer(21)
          call assoc$closer(22)
          if(itest .ne. 1) return
          write(6,1510)
1510      format(/" LEVEL OF INCOMPATIBILITY BETWEEN GRIDDED DATA SETS"
                 " PROHIBITS"/" GRID-TO-GRID OPERATIONS."/)
          call prompt("WHEN READY TO PROCEED, ENTER CARRIAGE RETURN.",45)
          read (5,1001) itest
          return 1
700       call assoc$closer(21)
          go to 100
800       call assoc$closer(22)
          go to 150
          end
```

```
      subroutine bndin2(x1,y1,xsav,ysav,nint,is,j,jl)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/     xb(20000,2),   yb(20000,2),   nch(2000,2),
                         noch(2),       kod(2000,2),   ipar(2),
                         icnd(2),       kap(15,2),     nptot(2)
      common /chain/     icf(100),      icl(100),      xint(101),
                         iptf(2000),    iptl(2000)
      dimension      x1(2),    y1(2),    x2(2),    y2(2),
                     xsav(50), ysav(50)
c
c     Given a line segment, [(x1(1),y1(1)),(x1(2),y1(2))], this
c     subroutine scans through all of the subchains belonging to
c     the slices containing the line segment to determine all of
c     the intersections that the boundary makes with the line
c     segment.  The coordinates of these intersections are stored
c     in the  xsav- , ysav- arrays for use in the calling program.
c
c     The intersection finding logic is similar to the logic
c     contained in subroutine  bndint .
c
          tol       =  .0005*(xint(2) - xint(1))
          isn       =  3 - is
          nint      =  0
          iflag     =  0
          ind1      =  1
          ind2      =  100
c
c     Determine the indices for the first (left), ind1 , and
c     the last (right), ind2 , slice encompassing the line
c     segment to be tested for boundary intersections.  Except
c     for the  x-interval defining the first slice which is
c     both left- and right-closed, all  x-intervals defining
c     the other slices are left-open and right-closed.
c
          do 100 i=1,100
          if((xint(i) - x1(1))*(x1(1) - xint(i+1)) .lt. 0.)  go to 50
          if(i .gt. 1  .and.  x1(1) .le. xint(i)) go to 50
          ind1      =  i
50        if((xint(i) - x1(2))*(x1(2) - xint(i+1)) .lt. 0.)  go to 100
          if(i .gt. 1  .and.  x1(2) .le. xint(i))  go to 100
          ind2      =  i
100       if(ind1 .ne. 1  .and.  ind2 .ne. 100)  go to 150
150       if(x1(1) .gt. xint(101))  ind1 = 100
          if(x1(2) .lt. xint(1)  )  ind2 = 1
```

```
          if(ind1 .le. ind2)  go to 200
          isav      =  ind2
          ind2      =  ind1
          ind1      =  isav
c
c     Begin testing the  is-th  boundary, slice by slice, to
c     find all intersections with line segment [(x1(1),y1(1)),
c     (x1(2),y1(2))], incrementing  ind1  for each slice until
c     it exceeds  ind2 .
c
200       if(ind1 .gt. ind2)  return
c
c     Determine the indices for the first,  ic1 , and the last,
c     ic2 , subchain in the  ind1-th  slice.
c
          ic1       =  icf(ind1)
          ic2       =  icl(ind1)
c
c     Begin testing the set of boundary subchains bridging
c     the  ind1-th  slice, incrementing  ic1  until it
c     exceeds  ic2 .
c
300       if(ic1 .gt. ic2)  go to 500
c
c     Determine the indices for the first,  ipt1 , and the last,
c     ipt2 , point in the  ic1-th  subchain.
c
          ipt1      =  iptf(ic1)
          ipt2      =  iptl(ic1)
          ic1       =  ic1 + 1
c
c     Scan through all the line segments in the  ic1-th
c     subchain, indexing the points in the subchain from
c     ipt1  to  ipt2 , and incrementing  ipt1  until it
c     exceeds  ipt2 .
c
c     If  ipt2  is negative, indicating that the subchain
c     was derived from one of the bottom boundaries added
c     to complete the set of boundary chains, there is no
c     need to test for an intersection and the test for
c     ipt1  greater than  ipt2  is properly passed.
c
          x2(1)     =  xb(ipt1,is)
          y2(1)     =  yb(ipt1,is)
400       if(ipt1 .ge. ipt2)  go to 300
          ipt1      =  ipt1 + 1
          x2(2)     =  xb(ipt1,is)
          y2(2)     =  yb(ipt1,is)
c
c     Test to determine if the boundary subchain segment
c     intersects with the given line segment.  If not,
c     iflag  is not equal to one, so skip to the next
c     segment of the subchain.
c
          call line2(x1,y1,x2,y2,xq,yq,iflag)
          if(iflag .ne. 1)  go to 450
c
c     If the second endpoint of the given line segment intersects
c     the boundary and it is not the last line segment of a
c     chain, then check a point  (xt,yt)  on the given line
```

```
c     segment as well as a similar point on the succeeding
c     line segment to determine if they are both on the same
c     side of the boundary chain (ino1 + ino2 .ne. 1), in
c     which case, do not append this intersection to the
c     xsav- , ysav- arrays..
c
          test1     =  abs(x1(2)-xq)
          test2     =  abs(y1(2)-yq)
          if(test1 + test2 .gt. tol  .or.  j .eq. jl)  go to 420
          xt        =  xb(j,isn) - .005*(xb(j,isn) - xb(j-1,isn))
          yt        =  yb(j,isn) - .005*(yb(j,isn) - yb(j-1,isn))
          call inside$inslic(is,ino1,xt,yt)
          xt        =  xb(j,isn) + .005*(xb(j+1,isn) - xb(j,isn))
          yt        =  yb(j,isn) + .005*(yb(j+1,isn) - yb(j,isn))
          call inside$inslic(is,ino2,xt,yt)
          if(ino1 + ino2 .ne. 1)  go to 450
420       if(nint .le. 0)  go to 440
c
c     Test the new intersection point against the intersection
c     points already stored in the  xsav- , ysav- arrays to
c     determine if it is a duplicate.  If it is a duplicate,
c     do not append it to these arrays..
c
          do 430 n=1,nint
          qnorm     =  abs(xq - xsav(n)) + abs(yq - ysav(n))
430       if(qnorm .le. tol)  go to 450
c
c     If the intersection point is acceptable, append its coordinates
c     onto the  xsav- , ysav- arrays and increment the intersection
c     counter,  nint .
c
440       nint      =  nint + 1
          xsav(nint)=  xq
          ysav(nint)=  yq
450       x2(1)     =  x2(2)
          y2(1)     =  y2(2)
          go to 400
500       ind1      =  ind1 + 1
          go to 200
          end
```

```
      subroutine bndint(xl1,yl1,xl2,yl2,xq,yq,iflag,is)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/     xb(20000,2),   yb(20000,2),   nch(2000,2),
                         noch(2),       kod(2000,2),   ipar(2),
                         icnd(2),       kap(15,2),     nptot(2)
      common /chain/     icf(100),      icl(100),      xint(101),
                         iptf(2000),    iptl(2000)
      dimension      x1(2),    y1(2),    x2(2),    y2(2)
      iflag      =  0
      x1(1)      =  xl1
      y1(1)      =  yl1
      x1(2)      =  xl2
      y1(2)      =  yl2
      ind1       =  1
      ind2       =  100
      do 100 i=1,100
      if((xint(i) - xl1)*(xl1 - xint(i+1)) .lt. 0.)  go to 50
      if(i .gt. 1  .and.  xl1 .le. xint(i)) go to 50
      ind1       =  i
50    if((xint(i) - xl2)*(xl2 - xint(i+1)) .lt. 0.)  go to 100
      if(i .gt. 1  .and.  xl2 .le. xint(i))  go to 100
      ind2       =  i
100   if(ind1 .ne. 1  .and.  ind2 .ne. 100)  go to 150
150   if(ind1 .le. ind2)  go to 200
      isav       =  ind2
      ind2       =  ind1
      ind1       =  isav
200   if(ind1 .gt. ind2)  return
      ic1        =  icf(ind1)
      ic2        =  icl(ind1)
300   if(ic1 .gt. ic2)  go to 500
      ipt1       =  iptf(ic1)
      ipt2       =  iptl(ic1)
      ic1        =  ic1 + 1
      x2(1)      =  xb(ipt1,is)
      y2(1)      =  yb(ipt1,is)
400   if(ipt1 .ge. ipt2)  go to 300
      ipt1       =  ipt1 + 1
      x2(2)      =  xb(ipt1,is)
      y2(2)      =  yb(ipt1,is)
      call line2(x1,y1,x2,y2,xq,yq,iflag)
      if(iflag .eq. 1)  return
      x2(1)      =  x2(2)
      y2(1)      =  y2(2)
```

```
            go to 400
500         ind1        =   ind1 + 1
            go to 200
            end
```

```
      subroutine bndpnt(xc,nc,ymin,ymax)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/     xb(20000,2),   yb(20000,2),   nch(2000,2),
                         noch(2),       kod(2000,2),   ipar(2),
                         icnd(2),       kap(15,2),     nptot(2)
      common /chain/     icf(100),      icl(100),      xint(101),
                         iptf(2000),    iptl(2000)
      dimension          x1(2),    y1(2),    x2(2),    y2(2)
      dimension          xc(4),    nc(4)
      common /bndpnt/    yp(4,50), in(4)
      y1(1)      =  ymin
      y1(2)      =  ymax
      is         =  2
      do 300 l=1,4
      x1(1)      =  xc(l)
      x1(2)      =  xc(l)
      nc(l)      =  0
      call insideSinslie(is,in(l),xc(l),ymin)
      ldex       =  1
      test       =  abs(xint(1) - xc(l))
      if(test .le. 0.)  go to 30
      do 20 k=1,100
      ldex       =  k
      test       =  (xint(k) - xc(l))*(xc(l) - xint(k+1))
      if(test .gt. 0.)  go to 30
      test       =  abs(xint(k+1) - xc(l))
      if(test .le. 0.)  go to 30
20    continue
      return
30    continue
      i1         =  icf(ldex)
      i2         =  icl(ldex)
      if(i1 .gt. i2)  go to 300
      do 200 i=i1,i2
      jb         =  iptf(i)
      jl         =  iptl(i)
      x2(1)      =  xb(jb,is)
      y2(1)      =  yb(jb,is)
      jbp        =  jb + 1
      do 200 j=jbp,jl
      x2(2)      =  xb(j,is)
      y2(2)      =  yb(j,is)
      call line2(x1,y1,x2,y2,xq,yq,iflag)
      if(iflag .ne. 1)  go to 100
```

```
      nc(l)      =  nc(l) + 1
      nn         =  nc(l)
      yp(l,nn)   =  yq
100   x2(1)      =  x2(2)
      y2(1)      =  y2(2)
200   continue
300   continue
      return
      entry inpnt(ym,yn,yi,nc,ii,inreg,iflg)
      iflg       =  0
      if(noch(is) .ne. 0)  go to 400
      inreg      =  1
      return
400   inreg      =  0
      if(nc(ii) .le. 0)  return
      lbelow     =  0
      int        =  0
      idx        =  nc(ii)
      do 500 i=1,idx
      if(yp(ii,i) .le. yn)  lbelow = lbelow + 1
500   if((ym-yp(ii,i))*(yp(ii,i)-yn) .ge. 0.)  int = i
      if(mod(lbelow+ipar(is),2) .eq. 1)  inreg = 1
      if(int .eq. 0)  return
      yi         =  yp(ii,int)
      iflg       =  1
      return
      end
```

```
      subroutine boundary
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /switch/     isobs,          idum2,iwind,itek,idum(6)
      common /corner/     xcorn(4),       ycorn(4)
      common /graphc/     xmin,           ymin,           zmin,
           zgrid(65000),  xmax,           ymax,           zmax,
           istat(65000),  nx,             ny,             nz,
           x(16000),      nxgd,           nygd,           ncell,
           y(16000),      delgrd,         igrid,          icell,
           into,          zint,           zlevel,         level,
           nbold,         iplot,          dashl,          pcrang,
           levmin,        levmax,         ifit,           kset,
           ndig,          ktour,          ismth,          icnt
      common /bound/      xb(20000,2),    yb(20000,2),    nch(2000,2),
                          noch(2),        kod(2000,2),    ipar(2),
                          icod(2),        kap(15,2),      nptot(2)
      common /files/      filnm1,         filnm2,         filnm3,
                          filgrd,         filobs
      character*8         filnm1,         filnm2,         filnm3,
                          filgrd,         filobs,
                          file1,          file2
      dimension           icom(10)
      data icom     /"valu","curs","comb","disp","reve",
                     "wind","unwi","dum3","dum4","quit"/
      call newpag
      call bell
      call anmode
      call prompt("PLEASE PRESS CARRIAGE RETURN: ",30)
      read(5,101) itest
      if(itest .ne. " ")  go to 100
      call newpag
      call anmode
      write(6,51)
51    format(/" When prompted with ENTER BOUNDARY COMMAND: ,",
            /" respond with one of the following commands to",
            /" manipulate or display the boundaries.")
      write(6,52)
52    format(/5x,"valueset",7x,"given a specified contour level, this command",
            /20x,"is used to create a boundary from the gridded data")
      write(6,53)
53    format(/5x,"cursor",9x,"this command permits the entry of a boundary",
            /20x,"from the graphics terminal by means of the cursor.")
      write(6,54)
54    format(/5x,"combine",8x,"this command permits the creation of a new",
```

```
              /20x,"boundary from the logical combination of two selected",
              /20x,"boundaries.")
            write(6,55)
55          format(/5x,"display",8x,"this command permits the  display of a selected",
              /20x,"boundary with a shading of the area to be included.")
            write(6,56)
56          format(/5x,"reverse",8x,"this command permits the reversal of the",
              /20x,"inclusion and the exclusion regions.")
            write(6,58)
58          format(/5x,"quit",11x,"this command terminates the boundary manipulation",
              /20x,"and display procedures.")
            write(6,59)
59          format(/5x,"Note that any of these commands may be executed by entering",
              /5x,"only the first four characters of the command."/)
            call prompt("WHEN READY TO PROCEED, ENTER CARRIAGE RETURN! ",46)
            read(5,101)  itest
100         call newpag
            call anmode
            call prompt("ENTER BOUNDARY COMMAND: ",24)
            read(5,101) komand
101         format(2a4)
            do 110 i=1,10
            igo       =  i
110         if(komand .eq. icom(i))  go to 120
            go to 100
120         if(igo .eq. 1)  go to 1000
            if(igo .eq. 10) go to 150
            ierr=1
            write(6,101)
            call prompt("DEFINE BOUNDARY FRAME: ",23)
            read(5,1661) file1
            if(file1.eq." ") go to 150
            if(file1.eq."input") go to 140
            call gridin$name(file1)
            if(iwind .eq. 1)  call window$windsv
            go to 150
140         write(6,101)
            write(6,141)
141         format(" AT THE FOLLOWING PROMPTS, ENTER THE MINIMUM"/
              " AND THE MAXIMUM X,Y-COORDINATES FOR USE IN"/
              " DEFINING THE PLOT WINDOW FOR THE BOUNDARY"/
              " DISLAY.")
            call bell
            call anmode
            call prompt("XMIN: ",6)
            read(5,142)  xmin
142         format(v)
            call bell
            call anmode
            call prompt("XMAX: ",6)
            read(5,142)  xmax
            call bell
            call anmode
            call prompt("YMIN: ",6)
            read(5,142)  ymin
            call bell
            call anmode
            call prompt("YMAX: ",6)
            read(5,142)  ymax
            xcorn(1)  =  xmin
```

```
      xcorn(2)  =  xmax
      xcorn(3)  =  xmax
      xcorn(4)  =  xmin
      ycorn(1)  =  ymin
      ycorn(2)  =  ymin
      ycorn(3)  =  ymax
      ycorn(4)  =  ymax
150   go to (1000,2000,3000,4000,5000,6000,7000,8000,9000,9999),igo
1000  call gridin
      is        =  2
      call window$unwind
      call conchn(is)
1500  call complt(is)
      do 1520 i=1,15
1520  kap(i,is) =  " "
      call newpag
      call anmode
1521  format(/" DO YOU WISH TO SAVE THIS BOUNDARY FILE?")
      write(6,1521)
1600  call prompt("(yes or no): ",13)
      read(5,101)  itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 1600
      if(itest .eq. "no")  go to 100
      filnm3=" "
      call chnin$chnout(is)
      go to 100
2000  is        =  2
      call create(is)
      go to 1500
3000  write(6,101)
      call prompt("NAME OF BOUNDARY FILE NO. 1: ",29)
      read(5,1661)  file1
1661  format(a8)
      write(6,101)
      call prompt("NAME OF BOUNDARY FILE NO. 2: ",29)
      read(5,1661)  file2
      is        =  1
      isn       =  2
      call scrin(is,file1,file2,$3300)
      call chnin$filein(is,file1,ierr)
      call chnin$filein(isn,file2,ierr)
      filnm1    =  file1
      filnm2    =  file2
      decode(file1,101)  kap(6,isn),kap(7,isn)
      decode(file2,101)  kap(13,isn),kap(14,isn)
      decode(file1,101)  kap(5,is),kap(7,is)
      decode(file2,101)  kap(13,is),kap(14,is)
      call chain(xmin,xmax,ymin,ymax)
      call scrin$scrout(is)
3300  call logic(is)
      decode(filnm3,101)  kap(1,is),kap(2,is)
      decode(filnm3,101)  kap(1,isn),kap(2,isn)
      if(filnm3 .ne. " ")  call chnin$chnout(isn)
      go to 100
4000  is        =  2
4100  write(6,101)
      call chnin(is,ierr)
      if(ierr .ge. 2)  go to 100
      call crtbeg(xmin,xmax,ymin,ymax)
      call pltchn(is)
```

```
            call shade(is)
            call movabs(1,3100/(3-itek)**2)
            call anmode
            read(5,101) itest
            go to 100
5000        is         =  2
            call chnin(is,ierr)
            ipar(is)   =  1 - ipar(is)
            call crtbeg(xmin,xmax,ymin,ymax)
            call pltchn(is)
            call shade(is)
            call movabs(1,3100/(3-itek)**2)
            call anmode
            read(5,101) itest
            go to 100
6000        call window
            go to 100
7000        call window$unwind
            go to 100
8000        go to 100
9000        go to 100
9999        return
            end
```

```
      subroutine calbeg
c
c           This subroutine was written and programmed by
c                          A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      double precision    filnam
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          isw1,
     8     ndig,          isw3,          isw4,          isw5
      common /corner/     xcorn(4),      ycorn(4)
      common /factor/     fact
      dimension           logo(2),       equate(5)
      character*4         itext(10,10),  ktext(10)
      data logo/"GARN","ET"/
      call plots(0,0,66)
      ndig    = abs(aint(alog10(xmax)+1.))
      lines   = 0
      call prompt("NAME OF TITLE/TEXT FILE: ",25)
      read(5,1001)filnam
      if(filnam .eq. "none") go to 7
      call assoc(67,filnam,"si  ")
1001  format(a8)
      do 5 i=1,10
      read(67,1003,end=6)(itext(i,j),j=1,10)
5     lines   = i
6     call assoc$closer(67)
1003  format(10a4)
7     xline   = lines
      xrang   = xmax - xmin
      yrang   = ymax - ymin
      if(fact .le. 0.)  fact = 1.
12    ntimes  = 1. + xrang*fact/30.
      if(ntimes .le. 1)  go to 15
      fact    = fact*.5
      go to 12
15    call plot(yrang*fact,0.,2)
      call plot(yrang*fact,xrang*fact,2)
      call plot(0.,xrang*fact,2)
      call plot(0.,0.,2)
      call plot((ycorn(4)-ymin)*fact,(xmax-xcorn(4))*fact,3)
      do 20 i=1,4
```

```
          call plot((ycorn(i)-ymin)*fact,(xmax-xcorn(i))*fact,2)
20        continue
c
c       THE FOLLOWING PLOT COMMANDS ARE USED TO DRAW THE "GARNET"
c       LOGO AND TO WRITE OUT THE TITLE OF THE PLOT AND LINES OF
c       DESCRIPTIVE TEXT ACCOMPANYING THE PLOT.
c
          xmid    =  .5*(xmin + xmax)*fact
          call plot(-.75,xmid+4.575,3)
          call plot(-.75,xmid-4.575,2)
          call plot(-2.35-xline*.3,xmid-4.575,2)
          call plot(-2.35-xline*.3,xmid+4.575,2)
          call plot(-.75,xmid+4.575,2)
          call newpen(2)
          call plot(-1.5,xmid+1.805,3)
          call plot(-1.815,xmid+1.26,2)
          call plot(-1.815,xmid-1.26,2)
          call plot(-1.5,xmid-1.805,2)
          call plot(-1.185,xmid-1.26,2)
          call plot(-1.185,xmid+1.26,2)
          call plot(-1.5,xmid+1.805,2)
          call symbol(-1.658,xmid+.945,.32,logo,-90.,6)
          call plot(-1.5,xmid-1.88,3)
          call plot(-1.5,xmid-2.35,2)
          call newpen(1)
          call plot(-.87,xmid-1.26,2)
          call newpen(2)
          call plot(-1.11,xmid-1.26,2)
          call plot(-.87,xmid-1.26,3)
          call newpen(1)
          call plot(-.87,xmid+1.26,2)
          call newpen(2)
          call plot(-1.11,xmid+1.26,2)
          call plot(-.87,xmid+1.26,3)
          call newpen(1)
          call plot(-1.5,xmid+2.35,2)
          call newpen(2)
          call plot(-1.5,xmid+1.88,2)
          call plot(-1.5,xmid+2.35,3)
          call newpen(1)
          call plot(-2.13,xmid+1.26,2)
          call newpen(2)
          call plot(-1.89,xmid+1.26,2)
          call plot(-2.13,xmid+1.26,3)
          call newpen(1)
          call plot(-2.13,xmid-1.26,2)
          call newpen(2)
          call plot(-1.89,xmid-1.26,2)
          call plot(-2.13,xmid-1.26,3)
          call newpen(1)
          call plot(-1.5,xmid-2.35,2)
          call newpen(2)
          hline   =  .28
          height  =  .21
          sumlin  =  0.
          do 60 i=1,lines
          sumlin = sumlin + hline
          do 50 j=1,10
50        ktext(j)  =  itext(i,j)
          call symbol(-2.35-sumlin,xmid+4.2,
```

```
                        height,xtext,-90.,40)
      if(i .eq. 3)  hline =.20
      if(i .ne. 2) go to 60
      call newpen(1)
      height  =  .15
60    continue
      return
      entry calplt(xx,yy,ip)
      ipp     =  ip
      call plot((yy-ymin)*fact,(xmax-xx)*fact,ipp)
      xsav    =  xx
      ysav    =  yy
      icount  =  0
      ipp     =  2
      dq      =  0.
      return
      entry calnum(xx,yy,h,fpn,alf,ndec)
      call number((yy-ymin)*fact,(xmax-xx)*fact,h,fpn,57.296*alf-90.,ndec)
      return
      entry calash(xx,yy)
      dl      =  dashl
      dx      =  xx - xsav
      dy      =  yy - ysav
      dr      =  sqrt(dx**2 + dy**2)
      xq      =  xsav + dq*dx/dr
      yq      =  ysav + dq*dy/dr
      dx      =  dl*dx/dr
      dy      =  dl*dy/dr
100   continue
      dt      =  sqrt((xq - xsav)**2 + (yq - ysav)**2)
      if(dt .gt. dr) go to 200
      call plot((yq-ymin)*fact,(xmax-xq)*fact,ipp)
      xq      =  xq + dx
      yq      =  yq + dy
      ipp     =  2 + mod(icount+1,2)
      icount  =  icount + 1
      go to 100
200   call plot((yy-ymin)*fact,(xmax-xx)*fact,ipp)
      dq      =  dt - dr
      xsav    =  xx
      ysav    =  yy
      return
      entry calend
      call plot(0.,0.,999)
      call assoc$closer(66)
      return
      end
```

```
      subroutine cclock(area,x,y,n)
c
c           This subroutine was written and programmed by
c                              A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.  All rights and privilege
c           RESOURCES DATA SYSTEM.
c           of use belong to the U. S. Geological Survey.
c
c
      dimension     x(n),          y(n)
c
c  THIS SUBROUTINE IS DESIGNED TO TEST THE POINTS DEFINING A
c  CLOSED POLYGON TO DETERMINE IF THE ORDER OF THE VERTICES
c  IS CLOCKWISE OR COUNTERCLOCKWISE.  IF THE VERTICES ARE IN
c  A CLOCKWISE ORDER, THE ARRAYS ARE SHIFTED TO PRODUCE A
c  COUNTERCLOCKWISE ORDER.
c  THE TECHNIQUE FOR TESTING THE ORDER OF THE VERTICES IS AS
c  FOLLOWS.  ASSUME THAT  A  IS A VECTOR CONNECTING ANY PAIR OF
c  VERTICES  (X(I),Y(I)) AND (X(I+1),Y(I+1)), WITH THE DIRECTION
c  TOWARD (X(I+1),Y(I+1)), THAT  B  IS ANOTHER VECTOR GOING FROM
c  (X(I+1),Y(I+1) TO (X(I+2),Y(I+2)).  THAT IS,
c        DX1   =  X(I+1) - X(I),
c        DY1   =  Y(I+1) - Y(I),
c        DX2   =  X(I+2) - X(I+1),
c        DY2   =  Y(I+2) - Y(I+1),
c   AND  A     =  (DX1,DY1),
c        B     =  (DX2,DY2).
c  THEN, IF VECTOR  A  IS ROTATED INTO VECTOR  B , THE PARALLELOGRAM
c  AREA,  PAREA , DEFINED BY THE TWO VECTORS CAN BE DETERMINED FROM
c  THE MAGNITUDE OF THEIR CROSS PRODUCT.  THE SIGN OF THE CROSS
c  PRODUCT IS POSITIVE IF VECTOR  A  IS ROTATED COUNTERCLOCKWISE
c  INTO VECTOR  B , AND IT IS NEGATIVE IF THE ROTATION IS CLOCKWISE.
c  THE SUM OF THE CROSS PRODUCTS DEFINED BY THE FIRST SEGMENT VECTOR
c  CROSSED ONTO THE VECTORS FORMED WITH EACH OF THE SUBSEQUENT
c  SEGMENTS DETERMINES THE TOTAL AREA DEFINED BY EACH OF THE VECTOR
c  CROSS PRODUCTS.  BY DIVIDING THIS TOTAL PARALLELOGRAM AREA BY
c  TWO, THE TOTAL AREA OF THE POLYGON IS DETERMINED.
c  THE SIGN OF THIS SUM INDICATES WHETHER THE TRAVERSAL OF THE
c  CIRCUMFERENCE OF THE POLYGON IS CLOCKWISE OR COUNTERCLOCKWISE.
c  IF THE VALUE OF  PAREA  IS POSITIVE, THE ORDER OF THE VERTICES
c  IS COUNTERCLOCKWISE AND CONTROL IS RETURNED TO THE CALLING
c  PROGRAM.  IF THE TOTAL VALUE IS NEGATIVE, THE ORDER IS CLOCKWISE
c  AND THE LOGIC IS ENTERED FOR REVERSING THIS ORDER TO A COUNTER-
c  CLOCKWISE ORDER.
c
      nm1    =  n - 1
      parea  =  0.
      dx1    =  x(1) - x(nm1)
      dy1    =  y(1) - y(nm1)
```

```
      subroutine chain(xmin,xmax,ymin,ymax)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /switch/ isobs,icrt,idum3,itek,isdum(6)
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
                          noch(2),       kod(2000,2),   ipar(2),
                          icnd(2),       kap(15,2),     notot(2)
      dimension       x1(2),    y1(2),    x2(2),    y2(2)
      dimension       dist(50),      xrsv(50),      yrsv(50)
c
c     This subroutine searches through the b-array (is = 2) to
c     separate those chains which lie inside the a-array (is = 1)
c     boundaries from those chains which lie outside the a-array
c     boundaries.  Then it searches through the a-array to separate
c     those chains lying within the b-array boundaries from those
c     chains lying outside of the b-array boundaries.
c
c     As the chains are separated into subchains, each subchain
c     is assigned a code value (kod) so that:
c
c          if the i-th chain of the a-array is contained within the
c                    b-boundary, then  kod(i,1) = 1
c          if the k-th chain of the b-array is contained within the
c                    a-boundary, then  kod(k,2) = 2
c          if the i-th chain of the a-array is outside of the
c                    b-boundary, then  kod(i,1) = 3
c          if the k-th chain of the b-array is outside of the
c                    a-boundary, then  kod(k,2) = 4
c
c     Also,
c          nch(k,is) gives the number of points contained in the k-th
c                    chain of the is-th array
c          noch(is) gives the total number of chains contained in
c                    the is-th array
c          ipar(is) sets the parity for inclusion or exclusion with the
c                    the is-th boundary.
c          nptot(is) is the total number of points used to define the
c                    is-th boundary
c          icnd(is) = 0, contours outside of the boundary will be blanked
c                   = 1, contours outside of the boundary will be dashed
c                   = 2, contours outside of the boundary will be solid
c
c     First, divide the b-array (is = 2) boundaries into subchains
c     that lie either inside, or outside, of the a-array boundary
c
```

```
          tol       =  .000005*sqrt((xmax-xmin)**2 + (ymax-ymin)**2)
          is        =  1
          icrt      =  1
          call crtpeg(xmin,xmax,ymin,ymax)
          icrt      =  0
          call czaxis(0)
c
c     Then shift all of the points in the arrays to
c     be subdivided into chains so that they occupy the
c     highest indexed portion of the array, leaving the
c     lowest indexed portion of the array available for
c     storage of the subdivided chains.
c
100       call snift(is)
          nci       =  nocn(is)
          iref      =  2000 - nci
c
c     Shift the values giving the number of points of
c     each chain into the highest indexed portion of the
c     nch-array, leaving the lowest indexed portion of the
c     array available for storing the number of points in
c     each of the subdivided chains.
c
          do 110 i=1,nci
          idx       =  iref + i
          nch(idx,is)  =  nch(i,is)
110       nch(i,is) =  0
c
c     ich  indexes the subchain that is currently being
c          created.
c
c     nsub  counts the number of points being stored
c          in the ich-th subchain.
c
c     npt  counts the total number of points being stored
c          in the new subchain arrays.
c
c     isn  indexes the other boundary array from
c          that indexed by  is.
c
c     jptb  indexes the first point of the i-th chain
c          in the is-th array.
c
c     jpto1  indexes the second point of the i-th chain
c          in the is-th array.
c
c     jptl  indexes the last point of the i-th chain
c          in the is-th array.
c
          ich       =  0
          npt       =  0
          isn       =  3 - is
          jptb      =  20001 - nptot(is)
          call slice(xmin,xmax,isn)
c
c     Scan through each chain (indexed by  i ) in the  is-th  array,
c     first of all determining if the first point of that chain is
c     inside of, or outside of, the  isn-th  boundary set.  Then find
c     all of the intersections of that chain with the  isn-th  set
c     to create a set of subchains for the  is-th  boundary set.
```

```
c       Assign the appropriate value of  kod  to each subchain.
c
            do 600 i=1,nci
            if(nch(i+iref,is) .lt. 0)  go to 601
            isav        =  i
            jptl        =  jptb + nch(i+iref,is) - 1
            jptb1       =  jptb + 1
            dx          =  xb(jptb1,is) - xb(jptb,is)
            dy          =  yb(jptb1,is) - yb(jptb,is)
            del         =  sqrt(dx**2 + dy**2)
            if(del .le. 0.)  go to 600
            xm          =  xb(jptb,is) + tol*dx/del
            ym          =  yb(jptb,is) + tol*dy/del
c
c       If the point, (xm,ym), is above the  ymin  value and between
c       the  xmin  and the  xmax  values for the frame, skip to the
c       logic for determining if the point is inside of, or outside of,
c       the boundary set.
c
            if((xmin-xm)*(xm-xmax) .ge. 0. .and. ym .ge. ymin)  go to 160
c
c       Otherwise, search through the chain until a point is found
c       that lies within the frame.
c
            xl          =  xm
            yl          =  ym
            jnd         =  jptb
130         jnd         =  jnd + 1
            if(jnd .gt. jptl)  go to 600
            xf          =  xl
            yf          =  yl
            xl          =  xb(jnd,is)
            yl          =  yb(jnd,is)
            xm          =  xf
            ym          =  yf
c
c       If the first point of the line segment does not lie above the
c       ymin  value, recompute a new value for (xf,yf), determined
c       from the intersection of the [(xf,yf),(xl,yl)] line segment
c       with the horizontal line through  ymin .
c
            if(ym .ge. ymin)  go to 150
            if((yf-ymin)*(ymin-yl) .lt. 0.)  go to 130
            ym          =  ymin
            xm          =  xl + (ym - yl)*(xf - xl)/(yf - yl)
c
c       If the first point of the line segment, or the adjusted point
c       at the  ymin  intersection, does not lie within the  xmin ,
c       xmax  bounds, recompute a new value for (xm,ym), determined
c       from the intersection of the [(xm,ym),(xl,yl)] line segment
c       with either the  xmin , or the  xmax  frame side bounds.
c
150         if((xmin - xm)*(xm - xmax)  .ge. 0.)  go to 160
            test1       =  (xm - xmin)*(xmin - xl)
            test2       =  (xm - xmax)*(xmax - xl)
            if(test1 .lt. 0.  .and.  test2 .lt. 0.)  go to 130
            if(xm .gt. xmax)  xm = xmax
            if(xm .lt. xmin)  xm = xmin
            ym          =  yl + (xm - xl)*(yf - yl)/(xf - xl)
160         call inside$inslic(isn,inout,xm,ym)
```

```
          ich       =  ich + 1
          kod(ich,is)=  5 - isn - 2*inout
c
c     Append the first point of the the subchain to the front end
c     of the boundary arrays, initialize the subchain point count,
c     nsub , and increment the total point count,  npt .
c
          nsub      =  1
          x1(1)     =  xb(jptb,is)
          y1(1)     =  yb(jpto,is)
          call movea(x1(1),y1(1))
          npt       =  npt + 1
          xb(npt,is)=  x1(1)
          yb(npt,is)=  y1(1)
          call movea(x1(1),y1(1))
          do 500 j=jptb1,jptl
          test1     =  10.
          test2     =  10.
c
c     Scan through each line segment in the i-th chain of the
c     is-th array and test to see if the line segment intersects
c     with a chain from the isn-th boundary array.  If there is
c     an intersection, subdivide the chain from the is-th array
c     at the intersection point and then change the  kod  value
c     for the subsequent subchain.
c
c     The first end point of each line segment from the is-th
c     array is stored in (x1(1),y1(1)) and the second end point
c     is stored in (x1(2),y1(2)).  They are transmitted through
c     the calling sequence to subroutine  bndin2  to determine if
c     that line segment intersects with a line segment from the
c     other boundary array.
c
          x1(2)     =  xb(j,is)
          y1(2)     =  yb(j,is)
          call bndin2(x1,y1,xrsv,yrsv,nint,isn,j,jptl)
c
c     Finished searching for an intersection with a line segment
c     of the isn-th boundary array.
c
          if(nint .le. 0)  go to 490
c
c     At least one intersection point (xrsv,yrsv) has been found for
c     the j-th line segment of the i-th chain from the is-th array.
c     The intersection points are sorted in the order of their
c     distance from the first point (x1(1),y1(1)) of the line segment.
c     Then each intersection point is appended into the low index
c     portion of the is-th array and a new  kod  value is assigned
c     to note that the chain has crossed a boundary belonging to the
c     isn-th array.
c
          do 410  n=1,nint
410       dist(n)   =  sqrt((xrsv(n)-x1(1))**2 + (yrsv(n)-y1(1))**2)
          if(nint .ge. 2)  call tsort(dist,xrsv,yrsv,nint)
          nn        =  nint
          nint      =  0
          do 420 n=1,nn
          if(dist(n) .le. 0.)  go to 420
          if(nint .eq. 0)  go to 415
          if(abs(dist(n)-dist(n-1)) .le. 0.)  go to 420
```

```
415         nint      =  nint + 1
            xrsv(nint)=  xrsv(n)
            yrsv(nint)=  yrsv(n)
420         continue
            if(nint .eq. 0)  go to 490
            do 480 n=1,nint
            nsub      =  nsub + 1
            npt       =  npt + 1
            xb(npt,is)=  xrsv(n)
            yb(npt,is)=  yrsv(n)
            if(kod(ich,is) .le. 2)  call drawa(xrsv(n),yrsv(n))
            if(kod(ich,is) .ge. 3)  call dasha(xrsv(n),yrsv(n),1)
            npt       =  npt + 1
            xb(npt,is)=  xrsv(n)
            yb(npt,is)=  yrsv(n)
450         nch(ich,is)=  nsub
            ich       =  ich + 1
            nsub      =  1
            xp        =  x1(2)
            yp        =  y1(2)
            if(n .ge. nint)  go to 460
            xp        =  xrsv(n+1)
            yp        =  yrsv(n+1)
460         xp        =  .5*(xb(npt,is) + xp)
            yp        =  .5*(yb(npt,is) + yp)
            call inside$inslic(isn,inout,xp,yp)
            kod(ich,is)=  5 - isn - 2*inout
480         continue
            test1     =  abs(x1(2)-xrsv(nint))
            test2     =  abs(y1(2)-yrsv(nint))
            if(test1 + test2 .le. tol)  go to 495
c
c     Finished scanning through all chains in the isn-th
c     array.
c
c     Store points from the higher index portion of the is-th
c     array into a subchain of the lowest index portion of the
c     array.  Then process next chain from the is-th array.
c
490         nsub      =  nsub + 1
            npt       =  npt + 1
            xb(npt,is)=  x1(2)
            yb(npt,is)=  y1(2)
            if(kod(icn,is) .le. 2)  call drawa(x1(2),y1(2))
            if(kod(ich,is) .ge. 3)  call dasha(x1(2),y1(2),1)
495         x1(1)     =  x1(2)
            y1(1)     =  y1(2)
500         continue
c
c     Finished scanning all line segments in the ich-th chain of
c     the is-th array.
c
            nch(ich,is)=  nsub
            if(test1 + test2 .gt. tol)  go to 600
            npt       =  npt - 1
            ich       =  ich - 1
600         jptb      =  jptl + 1
601         notot(is) =  npt
            noch(is)  =  ich
            if(is .ne. 1)  go to 700
```

```
          nochsv     =  noch(is)
          ichsv      =  ich
          if(isav .ge. nci)  go to 630
          isav       =  isav + 1
          do 620 i=isav,nci
          jptl       =  jptb + iabs(nch(i+iref,is)) - 1
          do 610 j=jptb,jptl
          npt        =  npt + 1
          xb(npt,is)=   xb(j,is)
610       yb(npt,is)=   yb(j,is)
          ich        =  ich + 1
          nch(ich,is)=  nch(i+iref,is)
620       jptb       =  jptl + 1
          noch(is)   =  ich
c
c     Next, divide the a-array (is=1) boundaries into subchains
c     that lie either inside, or outside, of the b-array boundary.
c     Return to statement 100 and repeat the logic with  is  set
c     to the value  2 .
c
630       is         =  2
          call czaxis(1)
          go to 100
c
c     The information in the arrays describing the a-boundary
c     is concatenated to the arrays describing the b-boundary.
c     Thus, the information for these two boundaries will be
c     available for the different logical combination operations.
c
700       call coinline
          do 800 i=1,npt
          idx        =  nptot(1) + i
          xb(idx,1)  =  xb(i,2)
800       yb(idx,1)  =  yb(i,2)
          do 900 i=1,ich
          idx        =  nochsv + i
          kod(idx,1)=   kod(i,2)
900       nch(idx,1)=   nch(i,2)
          noch(1)    =  noch(2) + nochsv
          nptot(1)   =  nptot(2) + nptot(1)
          call bell
          call movabs(1,3100/(itek-3)**2)
          call anmode
          read(5,1001)  itest
1001      format(a4)
          return
          end
```

```
      subroutine cnnin(is,ierr1)
c
c          This subroutine was programmed by
c                          R. J. Smith
c          to the specifications of A. C. Olson for use in
c          the GARNET interactive graphics system.  GARNET
c          was developed to perform resource mapping and
c          resource estimations for the   NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/ xb(20000,2),yb(20000,2),nch(2000,2),noch(2),
     kod(2000,2),ipar(2),icnd(2),kap(15,2),nptot(2)
      common /files/ filnm1,filnm2,filnm3,filgrd,filobs
      character*8 file,filnm1,filnm2,filnm3,filgrd,filobs,file2
      icall=1
3321  if(icall.eq.1) ierr1=0
      if(icall.eq.2) ierr2=0
      kap(15,1)="no"
      call prompt("BOUNDARY FILE NAME: ",20)
      read 110,file2
      if(file2.eq."none") return
      if(file2.eq." ".and.filnm3.eq." ") go to 3321
      kap(15,1)=" "
      if(icall.eq.1) ierr1=1
      if(icall.eq.2) ierr2=0
      if(file2.eq.filnm3) return
      if(file2.eq." ") return
      filnm3=file2
      go to 90
      entry filein(is,file,ierr2)
      icall=2
      file2=file
      ierr2=1
90    call assoc(42,file2,"si  ")
      read(42,200,end=160,err=160) (kap(i,is),i=1,15),nptot(is),noch(is),
     icnd(is),ipar(is)
      read(42,210,end=160,err=160) (nch(i,is),i=1,noch(is))
      read(42,220,end=160,err=160) (kod(i,is),i=1,noch(is))
      read(42,230,end=160,err=160) (xb(i,is),yb(i,is),i=1,nptot(is))
      call closer(42)
      if(kap(1,is).ne." ") return
      if(icall.eq.1) ierr1=2
      if(icall.eq.2) ierr2=2
      call complt(is)
      call newpag
      write(6,1001)
1001  format(/" THE RAW BOUNDARY FILE JUST"/
             " DISPLAYED WAS NOT LOWER"/
             " COMPLETE.  IT HAS BEEN"/
             " COMPLETED.")
      go to 100
```

```
      entry cnnout(is)
      if(filnm3.ne." ") go to 120
100   call prompt("NAME OF OUTPUT BOUNDARY FILE: ",30)
      read 110,filnm3
110   format(a8)
      decode(filnm3,115) kap(1,is),kap(2,is)
115   format(2a4)
120   call assoc(41,filnm3,"so  ")
      if(icnd(is) .ne. 2  .and.  icnd(is) .ne. 3)  icnd(is) = 1
      write(41,200) (kap(i,is),i=1,15),nptot(is),noch(is),
      icnd(is),ipar(is)
      write(41,210) (nch(i,is),i=1,noch(is))
      write(41,220) (kod(i,is),i=1,noch(is))
      write(41,230) (xo(i,is),yb(i,is),i=1,nptot(is))
      call closer(41)
      return
160   print 250,file
      call closer(42)
      go to 3321
200   format(15a4,4i5)
210   format(16i5)
220   format(80i1)
230   format(8f10.4)
250   format(" FILE ",a8," IS NOT A PROPER BOUNDARY FILE")
      end
```

```
      subroutine circle(x,y,istate)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      dimension      ix(12),        iy(12)
      data ix,iy/-3,-9,-12,-12,-9,-3,3,9,12,12,9,3,12,9,3,-3,
                 -9,-12,-12,-9,-3,3,9,12/
      call czaxis(1)
      call movea(x,y)
      call movrel(24,0)
      do 100 i=1,12
100   call drwrel(ix(i),iy(i))
      call movrel(-24,0)
      if(istate .eq. 0) return
      call movrel(36,36)
      call drwrel(-72,-72)
      call movrel(0,72)
      call drwrel(72,-72)
      call movrel(-36,36)
      return
      end
```

```
      subroutine cirint(x1,y1,x2,y2,a,b,r,xs1,ys1,xs2,ys2,int)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive grapnics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      if((x1 - x2)**2 .le. 0.) go to 100
      slope  =  (y2 - y1)/(x2 - x1)
      s2p1   =  slope**2 + 1.
      q      =  slope*x1 - y1 + b
      radcnd =  s2p1*r**2 - (q - slope*a)**2
      if(radcnd .le. 0.) go to 200
      root   =  sqrt(radcnd)
      adum   =  a + slope*q
      xs1    =  (adum - root)/s2p1
      ys1    =  y1 + slope*(xs1 - x1)
      xs2    =  (adum + root)/s2p1
      ys2    =  y1 + slope*(xs2 - x1)
      int    =  1
      return
      entry vlcir(x1,y1,x2,y2,a,b,r,xs1,ys1,xs2,ys2,int)
100   radcnd =  r**2 - (x1 - a)**2
      if(radcnd .le. 0.) go to 200
      xs1    =  x1
      xs2    =  x2
      root   =  sqrt(radcnd)
      ys1    =  b - sqrt(radcnd)
      ys2    =  b + sqrt(radcnd)
      int    =  1
      return
200   int    =  0
      return
      end
```

```
      subroutine cirobs(xx1,xx2,radrel,*)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /surfac/     xob(1500),     yob(1500),
     1    zt(1500),      zob(1500),     resid(1500),
     2    m,             n,             np,
     3    perd,          resum,         ntc,
     4    npc,           msq,           msq2,
     5    msq3,          msq4,          sig
      common /graphc/     xmin,          ymin,          zmin,
     1    zgrid(65000),  xmax,          ymax,          zmax,
     2    istat(65000),  nx,            ny,            nz,
     3    x(16000),      nxgd,          nygd,          ncell,
     4    y(16000),      delgrd,        igrid,         icell,
     5    into,          zint,          zlevel,        level,
     6    nbold,         lplot,         dashl,         pcrang,
     7    levmin,        levmax,        ifit,          isw1,
     8    isw2,          isw3,          isw4,          isw5
      common /cobs/       xp(200),       yp(200),       nob
      nob     =  0
      xbeg    =  xx1 - radrel
      xend    =  xx2 + radrel
      do 100 i=1,np
      if((xbeg - xob(i))*(xob(i) - xend) .lt. 0.)  go to 100
      nob     =  nob + 1
      xp(nob) =  xob(i)
      yp(nob) =  yob(i)
100   continue
      if(nob .eq. 0)  return1
      return
      entry ciry(yy1,yy2,radrel,*)
      ybeg    =  yy1 - radrel
      yend    =  yy2 + radrel
      do 200 i=1,nob
      if((ybeg - yp(i))*(yp(i) - yend) .gt. 0.)  return
200   continue
      return1
      entry cirtst(xt,yt,dmin,xxmin,yymin,radrel,*)
      dmin    =  2.*radrel
      kmin    =  0
      do 300 k=1,nob
      dist    =  sqrt((xp(k) - xt)**2 + (yp(k) - yt)**2)
      if(dist .ge. dmin)  go to 300
      kmin    =  k
      dmin    =  dist
```

```
300     continue
        if(kmin .le. 0) return 1
        xxmin   =  xp(kmin)
        yymin   =  yp(kmin)
        return
        end
```

```
      subroutine clevel
c
c
c           This subroutine was written and programmed by
c                              A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /index/ nx1,          nx2,          ny1,          ny2,
                     nxgm,         nygm,         xorig,        yorig
      common /graphc/      xmin,         ymin,         zmin,
           zgrid(65000),   xmax,         ymax,         zmax,
           istat(65000),   nx,           ny,           nz,
           x(16000),       nxgd,         nygd,         ncell,
           y(16000),       delgrd,       igrid,        icell,
           into,           zint,         zlevel,       level,
           nbold,          lplot,        dashl,        pcrang,
           levmin,         levmax,       ifit,         kset,
           ndig,           ktour,        ismth,        icnt
c
c*****************************************************************
c  FIND STARTING CONTOURS ALONG LEFT EDGE
c*****************************************************************
c
      klose   =  0
      do 100 i=ny1,nygm
      icell     =  (nx1 - 1)*nygd + i
      if(istat(icell) .gt. 0) go to 100
      igrid   =  (nx1 - 1)*ny + i
      z1      =  zgrid(igrid) - zlevel
      z2      =  zlevel - zgrid(igrid+1)
      if(z1*z2 .lt. 0.) go to 100
      if(abs(z1) .le. 0.)  go to 100
      into    =  1
      istat(icell)  =  4*into
      x(1)    =  xorig
      qiy     =  i - ny1
      qinc    =  .5
      if(abs(z1+z2) .gt. 0.)  qinc = z1/(z1 + z2)
      y(1)    =  yorig + delgrd*(qiy + qinc)
      call path(klose)
100   continue
c
c*****************************************************************
c  FIND STARTING CONTOURS ALONG TOP EDGE
c*****************************************************************
c
      do 200 i=nx1,nxgm
      icell   =  (i - 1)*nygd + nygm
```

```
          if(istat(icell) .gt. 0) go to 200
          igrid    =  (i - 1)*ny + nygm
          z1       =  zgrid(igrid+1) - zlevel
          z2       =  zlevel - zgrid(igrid+ny+1)
          if(z1*z2 .lt. 0.) go to 200
          if(abs(z1) .le. 0.)  go to 200
          into     =  2
          istat(icell)  =  4*into
          qix      =  i - nx1
          qinc     =  .5
          if(abs(z1+z2) .gt. 0.)  qinc = z1/(z1 + z2)
          x(1)     =  xorig + delgrd*(qix + qinc)
          qiy      =  ny2 - ny1
          y(1)     =  yorig + qiy*delgrd
          call path(klose)
200       continue
c
c*****************************************************************?
c  FIND STARTING CONTOURS ALONG RIGHT EDGE
c*****************************************************************
c
          do 300 i=ny1,nygm
          icell    =  nygd*(nxgm - 1) + i
          if(istat(icell) .gt. 0) go to 300
          igrid    =  ny*(nx2 - 2) + i
          z1       =  zgrid(igrid+ny+1) - zlevel
          z2       =  zlevel - zgrid(igrid+ny)
          if(z1*z2 .lt. 0.) go to 300
          if(abs(z1) .le. 0.)  go to 300
          into     =  3
          istat(icell)  =  4*into
          qix      =  nx2 - nx1
          x(1)     =  xorig + qix*delgrd
          qiy      =  i - ny1
          qinc     =  .5
          if(abs(z1+z2) .gt. 0.)  qinc = z2/(z1 + z2)
          y(1)     =  yorig + delgrd*(qiy + qinc)
          call path(klose)
300       continue
c
c*****************************************************************
c  FIND STARTING CONTOURS ALONG BOTTOM EDGE
c*****************************************************************
c
          do 400 i=nx1,nxgm
          icell    =  (i - 1)*nygd + ny1
          if(istat(icell) .gt. 0) go to 400
          igrid    =  (i - 1)*ny + ny1
          z1       =  zgrid(igrid+ny) - zlevel
          z2       =  zlevel - zgrid(igrid)
          if(z1*z2 .lt. 0.) go to 400
          if(abs(z1) .le. 0.)  go to 400
          into     =  4
          istat(icell)  =  4*into
          qix      =  i - nx1
          qinc     =  .5
          if(abs(z1+z2) .gt. 0.)  qinc = z2/(z1 + z2)
          x(1)     =  xorig + delgrd*(qix + qinc)
          y(1)     =  yorig
          call path(klose)
```

```
400       continue
c
c*********************************************************************
c  FIND STARTING POINTS FOR CLOSED CONTOURS
c*********************************************************************
c
          klose   =  1
          do 600 i=nx1,nxgm
          do 600 j=ny1,nygm
          icell   =  (i - 1)*nygd + j
          if(istat(icell) .gt. 0) go to 600
          igrid   =  (i - 1)*ny + j
          qiy     =  mod(igrid-1,ny) - ny1 + 1
          qix       =  igrid/ny - nx1 + 1
          z1      =  zgrid(igrid+1) - zlevel
          z2      =  zlevel - zgrid(igrid+ny+1)
          if(z1*z2 .le. 0.) go to 500
          into    =  2
          istat(icell)  =  4*into
          qinc    =  .5
          if(abs(z1+z2) .gt. 0.)   qinc = z1/(z1 + z2)
          x(1)    =  xorig + delgrd*(qix + qinc)
          y(1)    =  yorig + delgrd*(qiy + 1.)
          call path(klose)
          go to 600
500       if(i .eq. nxgm) go to 600
          z1      =  zgrid(igrid+ny+1) - zlevel
          z2      =  zlevel - zgrid(igrid+ny)
          if(z1*z2 .le. 0.) go to 600
          x(1)    =  xorig + delgrd*(qix + 1.)
          qinc    =  .5
          if(abs(z1+z2) .gt. 0.)  qinc = z2/(z1 + z2)
          y(1)    =  yorig + delgrd*(qiy + qinc)
          call path(klose)
600       continue
          return
          end
```

```
      subroutine coinline
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /corner/     xcorn(4),       ycorn(4)
      common /bound/      xb(20000,2),    yb(20000,2),    nch(2000,2),
                          noch(2),        kod(2000,2),    ipar(2),
                          icnd(2),        kap(15,2),      nptot(2)
c
c     This routine is designed to compare each chain in the first boundary
c     array, with all of the chains in the second boundary array.  If a
c     chain in the second boundary array is found which is identical to a
c     chain of the first boundary array, the  kod  value for the chain
c     from the second array is changed to  5 , if both boundary regions
c     lie on the same side of the boundary chain, or else the value
c     is changed to 6, if the two boundary regions lie on opposite sides
c     of the chain.  The  kod  value for the chain from the first
c     boundary array is reset to  0  since that chain is a duplicate
c     does not need to be used.
c
          tol       =  sqrt((xcorn(1)-xcorn(3))**2 + (ycorn(1)-ycorn(3))**2)
          tol       =  .00005*tol
          nc1       =  noch(1)
          nc2       =  noch(2)
          jb1       =  0
          do 300 i=1,nc1
          if(nch(i,1) .lt. 0)  go to 300
          ncni      =  nch(i,1)
          jo2       =  0
          do 200 k=1,nc2
          if(nch(k,2) .lt. 0)  go to 200
          if(ncni .ne. nch(k,2))  go to 200
          do 100 j=1,ncni
          test1     =  xb(jb1+j,1) - xb(jb2+j,2)
          test2     =  yb(jo1+j,1) - yb(jb2+j,2)
100       if(sqrt(test1**2 + test2**2) .gt. tol)  go to 200
          kod(i,1)  =  0
          jq        =  jb2 + 2
120       dx        =  xb(jq,2) - xb(jq-1,2)
          dy        =  yb(jq,2) - yb(jq-1,2)
          xc        =  .5*(xb(jq,2) + xb(jq-1,2))
          yc        =  .5*(yb(jq,2) + yb(jq-1,2))
          del       =  sqrt(dx**2 + dy**2)
          if(del .le. 0.)  go to 180
          x1        =  xc - 2.*tol*dy/del
          y1        =  yc + 2.*tol*dx/del
```

```
      call inside(1,inr1,x1,y1)
      if(inr1 .eq. 1)  go to 140
      call inside(2,inr1,x1,y1)
140   x1        =  xc + 2.*tol*dy/del
      y1        =  yc - 2.*tol*dx/del
      call inside(1,inr2,x1,y1)
      if(inr2 .eq. 1)  go to 160
      call inside(2,inr2,x1,y1)
160   kod(k,2)  =  4 + inr1 + inr2
      go to 200
180   if(jq .ge. jb2 + nch(k,2))  go to 200
      jq        =  jq + 1
      go to 120
200   jb2       =  jb2 + nch(k,2)
300   jb1       =  jb1 + nchi
      return
      end
```

```
      subroutine complt(is)
c
c          This subroutine was written and programmed by
c                         A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /switch/     isobs,          icrt,idum5,itek,isdum(6)
      common /complete/   icomp
      common /graphc/     xmin,           ymin,           zmin,
           zgrid(65000),  xmax,           ymax,           zmax,
           istat(65000),  nx,             ny,             nz,
           x(16000),      nxgd,           nygd,           ncell,
           y(16000),      delgrd,         igrid,          icell,
           into,          zint,           zlevel,         level,
           nbold,         lplot,          dashl,          pcrang,
           levmin,        levmax,         ifit,           kset,
           ndig,          ktour,          ismth,          icnt
      common /corner/     xcorn(4),       ycorn(4)
      common /bound/      xb(20000,2),    yb(20000,2),    nch(2000,2),
                          noch(2),        kod(2000,2),    ipar(2),
                          icnd(2),        kap(15,2),      nptot(2)
      common /files/      filnm1,         filnm2,         filnm3,
                          filgrd,         filobs
      character*8         filnm1,         filnm2,         filnm3,
                          filgrd,         filobs,         file
      dimension           xbel(20),       x1(2),          y1(2),
                          ybel(20),       x2(2),          y2(2)
c
c     Establish the corner points for the lower map edge.  If a
c     grid file is not used to define the corner points, the
c     xmax ,  xmin , and  ymin  values that have been specified
c     as prompted input will be used.
c
          icomp     =  0
          tol       =  .1*sqrt((xmax-xmin)**2 + (ymax-ymin)**2)
          xl        =  xmin
          yl        =  ymin
          xr        =  xmax
          yr        =  ymin
          do 20 i=1,4
          test      =  sqrt((xcorn(i)-xl)**2 + (ycorn(i)-yl)**2)
          if(test .gt. tol)  go to 10
          xl        =  xcorn(i)
          yl        =  ycorn(i)
          go to 20
10        test      =  sqrt((xcorn(i)-xr)**2 + (ycorn(i)-yr)**2)
          if(test .gt. tol)  go to 20
          xr        =  xcorn(i)
```

```
          yr        =  ycorn(i)
20        continue
c
c     Plot the boundary contained in the raw data file.
c
          nc        =  noch(is)
          if(icrt .eq. 1)  go to 40
          call crtbeg(xmin,xmax,ymin,ymax,itek)
          jl        =  0
          do 30 i=1,nc
          jb        =  jl + 1
          jl        =  jl + nch(i,is)
          call movea(xb(jb,is),yb(jb,is))
          jb        =  jb + 1
          do 30 j=jb,jl
30        call drawa(xb(j,is),yb(j,is))
c
c     Determine the minimum y-value in the  yb -array and save it
c     to use for the y-coordinate of the line segments in the
c     created lower boundary.
c
40        np        =  nptot(is)
c
c     Determine the parameters for describing the line bounding
c     the lower edge of the map.  Initialize the intersection
c     count,  int , to one, and the first x-coordinate for the
c     boundary intersection set,  xbel(1) , at the x-coordinate
c     value of the lower left corner point  xl .
c
          dydx      =  (yr - yl)/(xr - xl)
          int       =  1
          xbel(int) =  xl
          ybel(int) =  yl
c
c     Search through the boundary arrays to find those chains with
c     end points on or below the line bounding the lower map edge.
c     Store those points in the  xbel -array, with  int  providing
c     a count of the number of lower boundary intersections.
c
          jl        =  0
          do 200 i=1,nc
          jb        =  jl + 1
          jl        =  jl + nch(i,is)
          yy        =  yl + dydx*(xb(jb,is) - xl)
          if(yb(jb,is) .gt. yy)  go to 100
          int       =  int + 1
          xbel(int) =  xb(jb,is)
          ybel(int) =  yb(jb,is)
100       yy        =  yl + dydx*(xb(jl,is) - xl)
          if(yb(jl,is) .gt. yy)  go to 200
          int       =  int + 1
          xbel(int) =  xb(jl,is)
          ybel(int) =  yb(jl,is)
200       continue
          if(int .le. 1)  go to 400
          int       =  int + 1
          xbel(int) =  xr
          ybel(int) =  yl
c
c     After all intersection points have been determined, sort
```

```
c     the  xbel -array so that the values occur in ascending
c     order.
c
          call dsort(xbel,ybel,int)
c
c     Determine the new chains (line segments) lying along the
c     lower map edge, that are needed to complete the boundary
c     set.  Append their coordinates to the  (xb,yb) -array,
c     store the number of points (two) to the  nch -array, and
c     increment the boundary chain parameters,  noch  and  nptot .
c
          int         =  int - 1
          do 220 i=2,int,2
          call movea(xbel(i),ybel(i))
          np          =  np + 1
          xo(np,is) =  xbel(i)
          yb(np,is) =  ybel(i)
          call drawa(xbel(i+1),ybel(i+1))
          np          =  np + 1
          xb(np,is) =  xbel(i+1)
          yo(np,is) =  ybel(i+1)
          noch(is)  =  noch(is) + 1
          nc          =  noch(is)
          nch(nc,is)=  -2
220       continue
          notot(is) =  np
c
c     Use the cursor to identify the region of interest for use
c     in resource evaluation and mapping.  The input point is
c     tested and the value of  ipar  is set accordingly.
c
400       call bell
          call movabs(1,3100/(itek-3)**2)
          call anmode
          write(6,1002)
1002      format(" PLACE THE CURSOR IN THE DESIRED"/
                 " REGION.  ENTER AN ""i"" TO"/
                 " DESIGNATE THAT REGION FOR"/
                 " INCLUSION IN THE RESOURCE"/
                 " CALCULATIONS.")
          call stty("-modes","erkl,ctl_char")
          call vcursr(itest,xin,yin)
          call stty("-modes","erkl,ctl_char")
          if(itest .ne. 105)  go to 400
          ipar(is)  =  0
          call inside(is,inout,xin,yin)
          ipar(is)  =  1 - inout
          call shade(is)
          call anmode
          call bell
          read(5,1001)  itest
1001      format(a4)
          return
          end
```

```
      subroutine conchn(is)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          kset,
     8     ndig,          ktour,         ismth,         icnt
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
     1                    noch(2),       kod(2000,2),   ipar(2),
     2                    icnd(2),       kap(15,2),     nptot(2)
      common /corner/     xcorn(4),      ycorn(4)
      common /baud/       ibaud
      common /factor/     scale
      common /switch/ idum1,idum2,idum3,itek,idum5,idum6,idum7,idum8,idum9,idum10
c
c     Initialize the parameters for plotting a smoothed boundary
c     created from the gridded data by means of the contouring
c     routines.
c
          nbold   =  2
          level   =  1
          ismth   =  1
          scale   =  1.
1001      format(20a4)
c
c     The following statements set up a scratch file to write
c     out the boundary level chains from subroutine  CLEVEL.
c
          call assoc(21,"scratch ","so  ")
40        call prompt("ENTER Z-VALUE TO DEFINE BOUNDARY: ",34)
          read(5,1030,err=40)  zlevel
1030      format(v)
          do 50 i=1,nz
50        istat(i)  =  0
c
c     Establish the plot frame parameters and initialize the Tektronix
c     routines, the number of grid cells , and the number of significant
c     digits in the boundary label.
c
          diag      =  sqrt((xmax-xmin)**2 + (ymax-ymin)**2)
```

```
          xmin1     =  amin1(xcorn(1),xcorn(2),xcorn(3),xcorn(4))
          xmax1     =  amax1(xcorn(1),xcorn(2),xcorn(3),xcorn(4))
          ymin1     =  amin1(ycorn(1),ycorn(2),ycorn(3),ycorn(4))
          ymax1     =  amax1(ycorn(1),ycorn(2),ycorn(3),ycorn(4))
          delta     =  .025*diag
          xm1       =  xmin1 - delta
          xm2       =  xmax1 + delta
          ym1       =  ymin1 - delta
          ym2       =  ymax1 + delta
          call initt(ibaud/10)
          call crtbeg(xm1,xm2,ym1,ym2,2)
          nxgd      =  nx - 1
          nygd      =  ny - 1
          ncell     =  nxgd*nygd
          ndig      =  abs(aint(alog10(zlevel) + 1.))
c
c     Subroutine  CLEVEL  creates the boundary level chains
c     from the set of gridded data.  The value of  KLOSE
c     is  0  if the chain is open and  1  if the chain is
c     closed (i.e. a polygon).  All chains and polygons for the
c     specified  ZLEVEL  will be plotted.
c
          ktour     =  1
          call clevel(klose)
          call bell
          call movabs(1,3000/(itek-3)**2)
          call anmode
          write(6,1005)
1005      format(" WHEN READY TO CONTINUE,"/
                 " PRESS THE CARRIAGE RETURN.")
          read(5,1001)  itest
c
c     Clear the screen and initialize the plot parameters.  Reassign
c     the scratch file so that the chains and polygons may be read
c     in, in the same sequence as they were written onto the scratch
c     file from  CLEVEL .
c
          call newpag
          call assoc$closer(21)
          call assoc(22,"scratch ","si  ")
          call initt(ibaud/10)
          call czaxis(0)
c
c     Read the boundary chain coordinates from the file created
c     by the contouring routine and load them into the is-th
c     boundary array.  Plot these chains again onto the CRT display.
c
c          noc     indexes and counts the number of chains to
c                  to be stored.
c
c          np      indexes and counts the total number of points
c                  to be stored.
c
          noc       =  0
          np        =  0
100       read(22,1003,end=300) zlevel,icnt
1003      format(e15.7,i5)
1002      format(8f10.4)
          noc       =  noc + 1
          n1        =  np + 1
```

```
            np          =  np + icnt
            read(22,1002) (xb(i,is),yb(i,is),i=n1,np)
            ncn(noc,is)   =  icnt
            go to 100
300         noch(is)    =  noc
            nptot(is) =  np
            call assoc$closer(22)
            return
            end
```

```
      subroutine conin(*,klose,inout)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      double precision filnam
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          isw1,
     8     isw2,          isw3,          ismth,         icnt
      read(22,1001,end=100) zlevel,icnt,klose,inout
1001  format(e15.7,3i5)
      read(22,1002) (x(i),y(i),i=1,icnt)
1002  format(8f10.3)
      return
100   return 1
      entry ciopen
      call prompt("NAME OF CONTOUR INPUT FILE: ",30)
      read(5,1003) filnam
1003  format(a8)
      call assoc(22,filnam,"si  ")
      return
      entry ciclos
      call assoc$closer(22)
      return
      entry conout(klose,inout)
      write(21,1001) zlevel,icnt,klose,inout
      write(21,1002) (x(i),y(i),i=1,icnt)
      return
      entry coopen
      call prompt("NAME OF CONTOUR OUTPUT FILE: ",29)
      read(5,1003) filnam
      call assoc(21,filnam,"so  ")
      return
      entry coclos
      call assoc$closer(21)
      return
      end
```

```
      subroutine create(is)
c
c           This subroutine was written and programmed by
c                          A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /switch/ idum1,idum2,idum3,itek,idum4(6)
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     npold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          kset,
     8     ndig,          ktour,         ismth,         icnt
      common /corner/     xcorn(4),      ycorn(4)
      common /baud/       ibaud
      common /factor/     scale
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
     1                    noch(2),       kod(2000,2),   ipar(2),
     2                    icnd(2),       kap(15,2),     nptot(2)
      character*8         file
c
c
c     This subroutine is designed to permit the entry of boundaries
c     from the terminal by means of the cursor.  It permits the
c     display of another boundary for reference when creating the
c     new boundary.
c
c
c     iscl     is used to scale the message lines to suit the
c              addressable vertical pixcel range associated with
c              the specified model of Tektronix terminal.
c
c     lines    is used to adjust the cursor vertically for the
c              printing of message lines from the top to the
c              bottom of the scratch area of the screen.
c
      isn       =  3 - is
      iscl      =  (3-itek)**2
      lines     =  3100/iscl
1002  format(5a4)
100   write(6,1003)
1003  format(/" DO YOU WISH TO DISPLAY A BOUNDARY SET FOR"/
     1       " REFERENCE WHEN CREATING THE NEW BOUNDARY SET?")
120   call prompt("(yes or no): ",13)
      read(5,1002)  itest
```

```
          if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 120
          if(itest .eq. "no") go to 150
c
c     If it is desired to display a reference boundary to aid in the
c     creation of a set of boundary chains by means of the cursor,
c     the following statements provide for reading in the reference
c     boundary and then displaying it on the CRT display.
c
          call chnin(isn,ierr)
          call crtbeg(xmin,xmax,ymin,ymax)
          call czaxis(0)
          call pltchn(isn)
c
c     Begin the logic for entry of boundary chains by means
c     of the cursor.
c
c     noc       is used to index and count the total number of
c               chains entered.
c
c     np        is used to index and count the total number of
c               points entered into the boundary chain arrays.
c
150       if(itest .eq. "no")  call crtbeg(xmin,xmax,ymin,ymax)
          call czaxis(1)
          noc       =  0
          np        =  0
c
c     A new boundary chain is begun.
c
c     icnt      is used to index and count the total number of
c               points entered into the noc-th boundary chain.
c
200       noc       =  noc + 1
          icnt      =  0
c
c     Continue the entry of points into the noc-th boundary
c     chain.
c
220       icnt      =  icnt + 1
230       call bell
          call anmode
          call stty("-modes","^erkl,ctl_char")
          call vcursr(itest,xin,yin)
          call stty("-modes","erkl,^ctl_char")
          if(itest .ne. 120  .and.  itest .ne. 116)  go to 230
c
c     Plot the boundary chain point by point as it is input
c     by means of the cursor.  Then store the input points
c     in the boundary array.
c
          if(icnt .le. 1)  call pointa(xin,yin)
          if(icnt .gt. 1)  call drawa(xin,yin)
          np        =  np + 1
          xb(np,is) =  xin
          yb(np,is) =  yin
          if(itest .ne. 116)  go to 220
          nch(noc,is)=  icnt
          lines     =  lines - 200/iscl
          call movabs(1,lines)
          call anmode
```

```
      write(6,1004)
1004  format(" DO YOU WISH TO CREATE ANY"/
     *       " MORE BOUNDARIES?")
240   call prompt("(yes or no): ",13)
      read(5,1002) itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 240
      if(itest .eq. "yes")  go to 200
      noch(is)  =  noc
      nptot(is) = np
      return
      end
```

```
      subroutine cross(xx,yy)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          kset,
     8     ndig,          ktour,         ismth,         icnt
      common /factor/     fact
c
c*********************************************************************
c  THIS SUBROUTINE PLOTS A CROSS (A PLUS SIGN) WITH CENTER AT
c  COORDINATES  X  AND  Y .
c*********************************************************************
c
      if(fact .le. 0.)  fact = 1.
      dxx = .05/fact
      if(lplot .lt. 1) go to 100
      call calbeg$calplt(xx-dxx,yy,3)
      call calbeg$calplt(xx+dxx,yy,2)
      call calbeg$calplt(xx,yy-dxx,3)
      call calbeg$calplt(xx,yy+dxx,2)
      call calbeg$calplt(xx,yy,3)
100   if(lplot .gt. 1) go to 200
      call movea(xx,yy)
      call movrel(0,-10)
      call drwrel(0,20)
      call movrel(-10,-10)
      call drwrel(20,0)
      call movea(xx,yy)
200   return
      end
```

```
      subroutine crtbeg(xmin,xmax,ymin,ymax)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /corner/     xcorn(4),      ycorn(4)
      common /switch/     isobs,         icrt,          iwind,
           itek,          is5,           is6,           is7,
           is8,           is9,           is10
      iterm  =  2*itek - 1
      idum   =  (3 - itek)**2
      call term(iterm,4096/idum)
      call chrsiz(4)
      call newpag
      call logo
      ixmax  =  4096/idum - 1
      ixmin  =   976/idum - 1
      iymax  =  3120/idum - 1
      iymin  =  0
      call twindo(ixmin,ixmax,iymin,iymax)
      call movabs(ixmin,iymin)
      call drwabs(ixmax,iymin)
      call drwabs(ixmax,iymax)
      call drwabs(ixmin,iymax)
      call drwabs(ixmin,iymin)
      rangmx  =  amax1(xmax-xmin,ymax-ymin)
      x1  =  .5*(xmin + xmax - rangmx*1.01)
      x2  =  .5*(xmin + xmax + rangmx*1.01)
      y1  =  .5*(ymin + ymax - rangmx*1.01)
      y2  =  .5*(ymin + ymax + rangmx*1.01)
      call dwindo(x1,x2,y1,y2)
      call movea(xcorn(4),ycorn(4))
      do 100 i=1,4
100   call dasha(xcorn(i),ycorn(i),3)
      if(iwind .ne. 1)  return
      call movea(xmin,ymin)
      call dasha(xmax,ymin,1)
      call dasha(xmax,ymax,1)
      call dasha(xmin,ymax,1)
      call dasha(xmin,ymin,1)
      return
      end
```

```
      subroutine crtnum(x,y,val,it,ndrt)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.  All rights and privilege
c          of use belong to the U. S. Geological Survey.
c
c
      dimension      id(10),        ld(11)
      if(it .gt. 10) return
      do 10 i=1,11
10    ld(i)  =  0
      nbeg   =  2
      id(1)  =  45
      if(val .ge. 0.) nbeg = 1
      valp   =  abs(val)
      ld(1)  =  aint(valp*(10.**ndrt) + .5)
      ndig   =  nbeg
      do 100 i=2,11
      ld(i)  =  ld(i-1)/10
      ld(i-1)  =  ld(i-1) - 10*ld(i)
      ndig   =  ndig + 1
100   if(ld(i) .eq. 0) go to 200
200   k   =  ndig + 1
      do 300 i=1,ndig
      if(i .ne. ndrt + 1) go to 250
      k   =  k - 1
      id(k)  =  46
250   k   =  k - 1
      if(k .lt. nbeg) go to 400
300   id(k)  =  ld(i) + 48
400   call movea(x,y)
      call anstr(ndig,id)
      return
      end
```

```
      subroutine ctour
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /index/ nx1,          nx2,          ny1,          ny2,
                     nxgm,         nygm,         xorig,        yorig
      common /graphc/     xmin,         ymin,         zmin,
           zgrid(65000),  xmax,         ymax,         zmax,
           istat(65000),  nx,           ny,           nz,
           x(16000),      nxgd,         nygd,         ncell,
           y(16000),      delgrd,       igrid,        icell,
           into,          zint,         zlevel,       level,
           nbold,         lplot,        dashl,        pcrang,
           levmin,        levmax,       ifit,         kset,
           ndig,          ktour,        ismth,        icnt
      zm       =  .5*(zmax + zmin)
      do 20 i=1,nz
      zdel     =  amod(zgrid(i),zint)
      if(zdel .lt. .0001*zm)  zgrid(i)  =  zgrid(i) + .0001*zm
20    if(zint-zdel .lt. .0001*zm)  zgrid(i)  =  zgrid(i) - .0001*zm
      dzmax    =  amod(zmax,zint)
      dzmin    =  zint - amod(zmin,zint)
      levmax   =  aint((zmax - dzmax)/zint + .001)
      levmin   =  aint((zmin + dzmin)/zint + .001)
      do 800 lev=levmin,levmax
      level    =  lev
      do 10 i=nx1,nxgm
      do 10 j=ny1,nygm
      ind      =  (i-1)*nygd + j
10    istat(ind)=  0
      zlevel   =  level
      zlevel   =  zlevel*zint
      call clevel
c     if(kset .gt. 0)  call hachnd
800   continue
      return
      end
```

```
      subroutine curve(icount)
c
c           This subroutine was written and programmed by
c                           A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /graphc/      xmin,          ymin,          zmin,
           zgrid(65000),   xmax,          ymax,          zmax,
           istat(65000),   nx,            ny,            nz,
           x(16000),       nxgd,          nygd,          ncell,
           y(16000),       delgrd,        igrid,         icell,
           into,           zint,          zlevel,        level,
           nbold,          lplot,         dashl,         pcrang,
           levmin,         levmax,        ifit,          kset,
           ndig,           isw3,          ismth,         isw5
      common  /factor/ fact
      common /bound/       xb(20000,2),   yb(20000,2),   nch(2000,2),
                           noch(2),       kod(2000,2),   ipar(2),
                           icnd(2),       kap(15,2),     nptot(2)
      dimension            tx(2),         ty(2)
      dsum     =  0.
      dedge    =  0.
      scale    =  .5*sqrt((xmin - xmax)**2 + (ymin - ymax)**2)
      dmods1   =  0.
      dmods2   =  .12
      test     =  sqrt((x(1)-x(icount))**2 + (y(1)-y(icount))**2)
      ik       =  0
      inum     =  0
      ibrite   =  0
      ispace   =  0
      ii       =  0
      rangmx   =  amax1(xmax - xmin,ymax - ymin)
      radsv    =  .005*rangmx*fact
      if(ndig .le. 0)  ndig = 1
      xdig     =  ndig
      radial   =  .66*radsv*xdig/fact
      rrng     =  .05*rangmx
      x1       =  xmin + rrng
      x2       =  xmax - rrng
      y1       =  ymin + rrng
      y2       =  ymax - rrng
      if(mod(level,nbold) .eq. 0) ibrite  =  1
      if(lplot .le. 1  .or.  kset .lt. 0) call czaxis(ibrite)
      if(lplot .ge. 1) call newpen(ibrite+1)
      call dline(x(1),y(1),0,dashl)
      do 200 i=2,icount
      tx(1)      =  (xmin - x(i-1))*(x(i-1) - xmax)
      tx(2)      =  (xmin - x(i)  )*(x(i)   - xmax)
```

```
          ty(1)     =  (ymin - y(i-1))*(y(i-1) - ymax)
          ty(2)     =  (ymin - y(i)  )*(y(i)   - ymax)
c
c     If neither point of the line segment lies outside of the
c     x-range and the y-range of the frame, proceed to the logic
c     for normal processing.
c
          if(tx(1) .ge. 0. .and. tx(2) .ge. 0. .and.
             ty(1) .ge. 0. .and. ty(2) .ge. 0.)  go to 40
          if(tx(1) .lt. 0. .and. tx(2) .lt. 0.  .or.
             ty(1) .lt. 0. .and. ty(2) .lt. 0.)  go to 200
          xx        =  x(i-1)
          yy        =  y(i-1)
          itest     =  0
10        if((xmin - xx)*(xx - xmax) .ge. 0.)  go to 20
          if((x(i-1) - xmin)*(xmin - x(i)) .gt. 0.)  xx = xmin
          if((x(i-1) - xmax)*(xmax - x(i)) .gt. 0.)  xx = xmax
          yy        =  y(i-1) + (y(i) - y(i-1))*(xx - x(i-1))/(x(i) - x(i-1))
20        if((ymin - yy)*(yy - ymax) .ge. 0.)  go to 30
          if((y(i-1) - ymin)*(ymin - y(i)) .gt. 0.)  yy = ymin
          if((y(i-1) - ymax)*(ymax - y(i)) .gt. 0.)  yy = ymax
          xx        =  x(i-1) + (x(i) - x(i-1))*(yy - y(i-1))/(y(i) - y(i-1))
30        idrw      =  itest
          it1       =  itest + 1
c         if(tx(it1) .ge. 0. .and. ty(it1) .ge. 0.)  idrw = 1 - itest
          call dline(xx,yy,idrw,dashl,kset)
          xx        =  x(i)
          yy        =  y(i)
          itest     =  itest + 1
          if(itest .le. 1)  go to 10
          go to 200
40        delta     =  sqrt((x(i) - x(i-1))**2 + (y(i) - y(i-1))**2)
          dsum      =  dsum + delta
          dmod2     =  amod(dsum,.667)
          if(icount .le. 7) go to 110
          if(icount .le. 24  .and.  ismth .eq. 1)  go to 110
          if(ibrite .eq. 0) go to 110
          if((icount - i - 1)*(i - 2) .le. 0) go to 45
          dx1       =  x(i-2) - x(i-1)
          dy1       =  y(i-2) - y(i-1)
          dx2       =  x(i+1) - x(i)
          dy2       =  y(i+1) - y(i)
          dz        =  abs(dx1*dy2 - dx2*dy1)
          qnorm1    =  sqrt(dx1**2 + dy1**2)
          qnorm2    =  sqrt(dx2**2 + dy2**2)
          if(qnorm1*qnorm2 .le. 0.) go to 100
          sinth     =  dz/(qnorm1*qnorm2)
          if(sinth .gt. .5) go to 100
45        xtest     =  (x1 - x(i))*(x(i) - x2)
          ytest     =  (y1 - y(i))*(y(i) - y2)
          if(xtest .le. 0.  .or.  ytest .le. 0.) go to 100
          dedge     =  dedge + delta
          dmod1     =  amod(dedge,scale)
          if(dmod1 .lt. dmods1) inum = 0
          dmods1    =  dmod1
          if(inum .ne. 0) go to 100
          ispace    =  0
          inum      =  1
          ii        =  i
          call dline(x(i),y(i),1,dashl)
```

```
50          ispace  =  ispace + 1
            dx      =  x(i+ispace) - x(i)
            dy      =  y(i+ispace) - y(i)
            dist    =  sqrt(dx**2 + dy**2)
            if(dist .lt. radial) go to 50
            iii     =  ii + ispace
            dx      =  x(iii) - x(iii-1)
            dy      =  y(iii) - y(iii-1)
            dr      =  sqrt(dx**2 + dy**2)
            ratio   =  0.
            if(dr .gt. 0.) ratio = (dist - radial)/dr
            xx      =  x(iii) - dx*ratio
            yy      =  y(iii) - dy*ratio
            xm      =  .5*(x(i) + xx)
            ym      =  .5*(y(i) + yy)
            is      =  2
            if(kset .gt. 0)  call inside$inslic(is,inreg,xm,ym)
            if(icnd(is) .ne. 1 .or. inreg .eq. 1) go to 80
            ispace  =  0
            go to 100
80          angle   =  1.5708
            if(abs(xx-xm) .gt. 0.) angle = atan((yy - ym)/(xx - xm))
            xmm     =  xm - .35*radial*cos(angle)
            ymm     =  ym - .4*radial/xdig
            if(lplot .le. 1) call crtnum(xmm,ymm,zlevel,2,0)
            if(lplot .lt. 1) go to 90
            xc      =  radial*(.384*cos(angle) - .08*sin(angle))
            yc      =  radial*(.384*sin(angle) + .08*cos(angle))
            call calbeg$calnum(xm-xc,ym-yc,.48*radsv,zlevel,angle,-1)
90          call dline(xx,yy,0,dashl)
100         if(i .lt. ii + ispace) go to 200
110         call dline(x(i),y(i),1,dashl)
            if(dmod2 .gt. dmods2) go to 200
            if(abs(test) .ge. .001*radsv) go to 200
            delr    =  dmod2 + .667 - dmods2
            if(kset .lt. 0)  go to 200
c           call hachur(x(i),y(i),x(i-1),y(i-1),delr,ik)
200         dmods2  =  dmod2
            return
            end
```

```
      subroutine decomp(a,ul,nm,na)
c
c  COMPUTES TRIANGULAR MATRICES  L  AND  U  AND PERMUTATION
c  MATRIX  P  SO THAT  LU =  PA.  STORES  L-I  AND  U  IN  UL.
c  ARRAY  IPS  CONTAINS PERMUTED ROW INDICES.
c
      dimension      a(na,1),      ul(na,1),      scales(100),
                     ips(100)
      common /ir/ ips
      common /iabort/ lost
      n       =  nm
c
c  INITIALIZE  IPS ,  UL ,  AND  SCALES.
c
      do 5 i=1,n
      ips(i)  =  i
      rownrm  =  0.
      do 2 j=1,n
      ul(i,j) =  a(i,j)
      if(rownrm - abs(ul(i,j)))  1,2,2
1     rownrm  =  abs(ul(i,j))
2     continue
      if(rownrm)  3,4,3
3     scales(i)  =  1./rownrm
      go to 5
4     call sing(1)
      scales(i)  =  0.
5     continue
c
c  GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
c
      nm1     =  n - 1
      do 17 k=1,nm1
      big     =  0.
      do 11 i=k,n
      ip      =  ips(i)
      size    =  abs(ul(ip,k))*scales(ip)
      if(size - big)  11,11,10
10    big     =  size
      idxpiv  =  i
11    continue
      if(big)  13,12,13
12    call sing(2)
      go to 17
13    if(idxpiv - k)  14,15,14
14    j       =  ips(k)
      ips(k)  =  ips(idxpiv)
      ips(idxpiv)  =  j
15    kp      =  ips(k)
      pivot   =  ul(kp,k)
      kp1     =  k + 1
      do 16 i=kp1,n
```

```
            ip      =  ips(i)
            em      = -ul(ip,k)/pivot
            ul(ip,k)= -em
            do 16 j=kp1,n
            ul(ip,j)=  ul(ip,j) + em*ul(kp,j)
16          continue
17          continue
            kp      =  ips(n)
            if(ul(kp,n))  19,18,19
18          call sing(2)
            lost    =  0
19          return
            end
```

```
      subroutine dline(x,y,idraw,dashl)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
c
c*****************************************************************
c   IDRAW   =  0 ... MOVE CURSOR TO X,Y POSITION WITHOUT LINE TRACE.
c          =  1 ... DRAW LINE TRACE TO X,Y POSITION.
c   ICOND   =  1 ... EXTERNAL LINES WILL BE BLANKED.
c          =  2 ... EXTERNAL LINES WILL BE DASHED.
c          =  3 ... EXTERNAL LINES WILL BE SOLID.
c   INREG   =  0 ... POINT LIES OUTSIDE REQUESTED REGION.
c          =  1 ... POINT LIES INSIDE REQUESTED REGION.
c   IFLAG   =  0 ... LINE DOES NOT INTERSECT REGION BOUNDARY.
c          =  1 ... LINE INTERSECTS REGION BOUNDARY.
c*****************************************************************
c
      common /graphc/     xmin,          ymin,          zmin,
           zgrid(65000),  xmax,          ymax,          zmax,
           istat(65000),  nx,            ny,            nz,
           xa(16000),     nxgd,          nygd,          ncell,
           ya(16000),     delgrd,        igrid,         icell,
           into,          zint,          zlevel,        level,
           nbold,         lplot,         dashl,         pcrang,
           levmin,        levmax,        ifit,          kset,
           ndig,          ktour,         ismth,         icnt
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
                          noch(2),       kod(2000,2),    ipar(2),
                          icnd(2),       kap(15,2),     nptot(2)
      is        =  2
      inreg     =  1
c
c     If a boundary set has been specified, determine if the
c     point (x,y) is inside the boundary.
c
          if(kap(15,1) .ne. "no")  call inside$inslic(is,inreg,x,y)
c
c     If a move has been specified, save the point coordinates
c     and related parameters.
c
          if(idraw .eq. 0)  go to 600
c
c     If this is the first point with a draw command, perform the
c     move on the saved point and then proceed through the draw
c     logic.
c
```

```
          if(idraw .eq. 1  .and.  idrwsv .eq. 0)  go to 700
c
c     If the option has been selected so that lines outside of the
c     boundary are solid, draw a solid line and return.
c
100       if(icnd(is) .ne. 3)  go to 200
          call draws(x,y)
          return
c
c     If the line segment lies totally within the boundary, proceed
c     to a normal draw operation.
c
200       if(inreg + inrgsv .eq. 2)  go to 500
c
c     If the line segment is partially within the boundary, determine
c     the intersection point and process accordingly.
c
          if(inreg + inrgsv .eq. 1)  go to 300
c
c     If the line segment is totally outside the boundary, dash it if
c     the dashed line option has been selected, or else do nothing
c     if the blanking option has been chosen.
c
          if(icnd(is) .eq. 2)  call draws$dash(x,y)
          go to 600
c
c     The line segment crosses over the boundary.  Determine the
c     intersection point (xr,yr) by scanning through the boundary
c     subchains in the is-th array.
c
300       call ondint(xsav,ysav,x,y,xr,yr,iflag,is)
c
c     If there is no intersection (iflag = 0), then the new value of
c     inreg  must be incorrect.  Change it and recycle through the
c     logic to select the appropriate draw sequence.
c
          if(iflag .eq. 1)  go to 350
          inreg     =  1 - inreg
          go to 200
c
c     If the line segment exits from the bounded region, draw to the
c     intersection and then process the remainder of the line according
c     to the option selected.
c
350       if(inreg .eq. 1)  go to 400
          call draws(xr,yr)
          if(icnd(is) .eq. 2)  call draws$dash(x,y)
          go to 600
c
c     If the line segment enters the bounded region, proceed to the
c     intersection according to the option selected and then draw
c     a solid line into the bounded region.
c
400       if(icnd(is) .eq. 1)  call draws$move(xr,yr)
          if(icnd(is) .eq. 2)  call draws$dash(xr,yr)
500       call draws(x,y)
c
c     Save the information parameters for use with the next
c     draw operation.
c
```

```
600       inrgsv    =  inreg
          idrwsv    =  idraw
          xsav      =  x
          ysav      =  y
          return    .
c
c     Execute move to saved point when  idraw  goes from  0  to  1 .
c
700       call draws3move(xsav,ysav)
          if(icnd(is) .eq. 3)  go to 500
          go to 200
          end
```

```
      subroutine dmatin(n,din)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      implicit double precision (a-h,o-z)
      dimension      din(69,69),    lc(69),        lr(69)
      do 801 i=1,n
      lc(i)   =  i
801   lr(i)   =  i
      do 810 i=1,n
      b       =  0.d0
      do 803 k=i,n
      do 803 j=i,n
      if(dabs(din(k,j)) - b)  803,802,802
802   kl      =  k
      jl      =  j
      b       =  dabs(din(k,j))
803   continue
      if(b)  701,701,804
804   j       =  lr(i)
      lr(i)   =  lr(jl)
      lr(jl)  =  j
      k       =  lc(i)
      lc(i)   =  lc(kl)
      lc(kl)  =  k
      do 805 j=1,n
      b       =  din(i,j)
      din(i,j)=  din(kl,j)
805   din(kl,j)  =  b
      do 806 k=1,n
      b       =  din(k,i)
      din(k,i)=  din(k,jl)
806   din(k,jl)  =  b
      b       =  din(i,i)
      din(i,i)   =  1.d0
      do 807 j=1,n
807   din(i,j)   =  din(i,j)/b
      do 810 k=1,n
      if(k-i)  808,810,808
808   b       =  din(k,i)
      din(k,i)   =  0.d0
      do 809 j=1,n
809   din(k,j)   =  din(k,j) - b*din(i,j)
810   continue
      do 814 i=1,n
811   i1      =  lr(i)
```

```
      if(i1-i)  812,814,812
812   do 813 j=1,n
      b       =  din(i,j)
      din(i,j)  =  din(i1,j)
813   din(i1,j)  =  b
      lr(i)   =  lr(i1)
      lr(i1)  =  i1
      go to 811
814   continue
      do 818 j=1,n
815   j1      =  lc(j)
      if(j-j1)  816,818,816
816   do 817 i=1,n
      b       =  din(i,j)
      din(i,j)   =  din(i,j1)
817   din(i,j1)  =  b
      lc(j)   =  lc(j1)
      lc(j1)  =  j1
      go to 815
818   continue
      return
701   print 2013,lc(kl)
2013  format(1h0,4x7h*** row,i3,32h is redundant ... call exit.***  )
      stop
      end
```

```
      subroutine draws(xx,yy)
c
c           This subroutine was written and programmed by
c                           A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
           zgrid(65000),  xmax,          ymax,          zmax,
           istat(65000),  nx,            ny,            nz,
           x(16000),      nxgd,          nygd,          ncell,
           y(16000),      delgrd,        igrid,         icell,
           into,          zint,          zlevel,        level,
           nbold,         lplot,         dashl,         pcrang,
           levmin,        levmax,        ifit,          kset,
           ndig,          ktour,         ismth,         icnt
      if(lplot .le. 1) call drawa(xx,yy)
      ip      =  2
      if(lplot .ge. 1) call calbeg$calplt(xx,yy,ip)
      return
      entry dasn(xx,yy)
      if(lplot .le. 1) call dasha(xx,yy,3)
      if(lplot .ge. 1) call calbeg$caldsh(xx,yy)
      return
      entry move(xx,yy)
      if(lplot .le. 1) call movea(xx,yy)
      ip      =  3
      if(lplot .ge. 1) call calbeg$calplt(xx,yy,ip)
      return
      end
```

```
      subroutine dsort(xq,dq,n)
c
c          This subroutine was written and programmed by
c                              A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      dimension xq(n),          dq(n)
      nm        =  n - 1
      do 200 i=1,nm
      xs        =  xq(i)
      js        =  i
      ip        =  i + 1
      do 100 j=ip,n
      if(xq(j) .ge. xs)  go to 100
      xs        =  xq(j)
      js        =  j
100   continue
      if(js .le. i)  go to 200
      xq(js)    =  xq(i)
      xq(i)     =  xs
      ds        =  dq(js)
      dq(js)    =  dq(i)
      dq(i)     =  ds
200   continue
      return
      end
```

```
      subroutine editor
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /surfac/     xob(1500 ),     yob(1500 ),
     1    zt(1500 ),     zob(1500 ),     residu(1500 ),
     2    m,             n,              np,
     3    perd,          resum,          ntc,
     4    npc,           msq,            msq2,
     5    msq3,          msq4,           sig
      common /graphc/     xmin,           ymin,           zmin,
     1    zgrid(65000),  xmax,           ymax,           zmax,
     2    istat(65000),  nx,             ny,             nz,
     3    x(16000),      nxgd,           nygd,           ncell,
     4    y(16000),      delgrd,         igrid,          icell,
     5    into,          zint,           zlevel,         level,
     6    nbold,         lplot,          dashl,          pcrang,
     7    levmin,        levmax,         ifit,           isw1,
     8    isw2,          ktour,          ismth,          icnt
      dimension    lpt(1500)
      xrng     =  xmax - xmin
      yrng     =  ymax - ymin
      rngmx    =  amax1(xrng,yrng)
      stndrd   =  .005*xrng
      call nome
      call anmode
      lines    =  3040
      do 50 i=1,np
50    lpt(i)   =  0
      ierr     =  3
5000  err      =  ierr
      err      =  err*sig
      call movabs(0,lines)
      call anmode
      write(6,1002) ierr
1002  format(1x,"VALUES EXCEED ",i2/1x,"TIMES ""RMS"" ERROR.")
      lines    =  lines - 110
      call movabs(243,0)
      do 100 i=1,np
      if(abs(residu(i)) .le. err) go to 100
      if(lpt(i) .ne. 0) go to 100
      lpt(i)   =  1
      xtest    =  (xmin - xob(i))*(xob(i) - xmax)
      ytest    =  (ymin - yob(i))*(yob(i) - ymax)
      if(xtest .le. 0.  .or.  ytest .le. 0.) go to 100
      call cross(xob(i),yob(i))
```

```
      yy       =  yob(i) - .02*yrng
      xx       =  xob(i) - .002*xrng
      call crtnum(xx,yy,zob(i),5,0)
100   continue
200   call bell
      call stty("-modes","^erkl,ctl_char")
      call vcursr(itest,xx,yy)
      call stty("-modes","erkl,^ctl_char")
      if(itest .eq. 110) go to 600
      do 300 i=1,np
      ii       =  i
      test     =  sqrt((xob(i) - xx)**2 + (yob(i) - yy)**2)
      if(test .le. stndrd)  go to 400
300   continue
      go to 200
400   if(itest .ne. 100) go to 500
      lpt(ii)  =  - lpt(ii)
      call circle(xob(ii),yob(ii),1)
      go to 200
500   if(itest .ne. 99) go to 200
      call movabs(0,lines)
      call anmode
      call prompt("VALUE: ",7)
      read(5,1001)value
1001  format(v)
      lines    =  lines - 55
      call movabs(975,0)
      zob(ii)  =  value
      call circle(xob(ii),yob(ii),0)
      go to 200
600   continue
      call bell
      ierr     =  ierr - 1
      if(ierr .ge. 2) go to 5000
      iii      =  0
      do 700 i=1,np
      if(lpt(i) .lt. 0) go to 700
      iii      =  iii + 1
      xob(iii)  =  xob(i)
      yob(iii)  =  yob(i)
      zob(iii)  =  zob(i)
      residu(iii)  =  residu(i)
700   continue
      return
      end
```

```
      subroutine gridin
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      character*8 filnm1,filnm2,filnm3,filgrd,filobs,filnam
      common /files/ filnm1,filnm2,filnm3,filgrd,filobs
      common /graphc/     xmin,          ymin,          zmin,
     1    zgrid(65000),   xmax,          ymax,          zmax,
     2    istat(65000),   nx,            ny,            nz,
     3    x(16000),       nxgd,          nygd,          ncell,
     4    y(16000),       delgrd,        igrid,         icell,
     5    into,           zint,          zlevel,        level,
     6    noold,          lplot,         dashl,         pcrang,
     7    levmin,         levmax,        ifit,          isw1,
     8    isw2,           ktour,         ismth,         icnt
      common /corner/     xcorn(4),      ycorn(4)
10    call prompt("NAME OF GRIDDED INPUT FILE: ",28)
      read(5,1001)filnam
1001  format(a8)
      entry name(filnam)
      if(filnam.eq." ".and.filgrd.eq." ") go to 10
      if(filnam.eq.filgrd) return
      if(filnam.eq." ") return
      call assoc(22,filnam,"si  ")
      read(22,1004,err=20) nx,ny,nz,xmin,xmax,ymin,ymax,zmin,zmax,delgrd,
     1          pcrang,xcorn,ycorn
1004  format(3i5/8f10.4/8f10.4)
      read(22,1005,err=20) (zgrid(i),i=1,nz)
1005  format(5e15.7)
      call assoc$closer(22)
      filgrd=filnam
      return
      entry grido
      call prompt("FILE NAME TO STORE GRID VALUES: ",32)
      read(5,1001)filnam
      call assoc(21,filnam,"so  ")
      write(21,1004) nx,ny,nz,xmin,xmax,ymin,ymax,zmin,zmax,delgrd,
     1          pcrang,xcorn,ycorn
      write(21,1005) (zgrid(i),i=1,nz)
      call assoc$closer(21)
      filgrd=filnam
      return
20    call assoc$closer(22)
      go to 10
      end
```

```
      subroutine gridit(c)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      dimension       c(69)
      common /surfac/      xob(1500 ),      yob(1500 ),
     zt(1500 ),       zob(1500 ),      residu(1500 ),
     m,               n,               np,
     perd,            resum,           ntc,
     npc,             msq,             msq2,
     msq3,            msq4,            sig
      common /graphc/      xmin,            ymin,            zmin,
      zgrid(65000),    xmax,            ymax,            zmax,
      istat(65000),    nx,              ny,              nz,
      x(16000),        nxgd,            nygd,            ncell,
      y(16000),        delgrd,          igrid,           icell,
      into,            zint,            zlevel,          level,
      nbold,           lplot,           dashl,           pcrang,
      levmin,          levmax,          ifit,            isw1,
      isw2,            ktour,           ismth,           icnt
      npc      =  (n+1)*(n+2)/2 - 1
      nctot    =  ntc + npc + 1
      nx       =  (xmax - xmin)/delgrd + 1.9999
      ny       =  (ymax - ymin)/delgrd + 1.9999
      nz       =  nx*ny
      zmax     =  -1.e10
      zmin     =   1.e10
      do  800 j=1,nx
      xx       =  j - 1
      xx       =  xx*delgrd + xmin
      do  800 i=1,ny
      yy       =  i - 1
      yy       =  yy*delgrd + ymin
      index    =  (j - 1)*ny + i
      zgrid(index)  =  c(nctot)
      if(m .le. 0) go to 550
      do  500 ii=1,m
      qi       =  ii
      im       =  (ii-1)*m
      qipd     =  qi*perd
      sx       =  sin(qipd*xx)
      cx       =  cos(qipd*xx)
      syi      =  sin(qipd*yy)
      cyi      =  cos(qipd*yy)
      do 400 jj=1,m
      qj       =  jj
```

```
      qjpd    =  qj*perd
      sy      =  sin(qjpd*yy)
      cy      =  cos(qjpd*yy)
      im1     =  im  + jj
      im2     =  im1 + msq
      im3     =  im1 + msq2
      im4     =  im1 + msq3
      zgrid(index)  =  zgrid(index) + c(im1)*sx*sy + c(im2)*sx*cy
                                    + c(im3)*cx*sy + c(im4)*cx*cy
400   continue
      mm1     =  msq4 + ii
      mm2     =  mm1 + m
      mm3     =  mm2 + m
      mm4     =  mm3 + m
      zgrid(index)  =  zgrid(index) + c(mm1)*sx  + c(mm2)*cx
                                    + c(mm3)*syi + c(mm4)*cyi
500   continue
550   continue
      if(n .le. 0) go to 800
      nsum    =  ntc
      do 700 jj=1,n
      nn      =  n + 2 - jj
      do 600 ii=1,nn
      tx      =  xx
      ty      =  yy
      ix      =  ii - 1
      jy      =  nn - ii
      if(ix .eq. 0)  tx = 1.
      if(jy .eq. 0)  ty = 1.
      term    =  tx**ix*ty**jy
      zgrid(index)  =  zgrid(index) + c(nsum+ii)*term
600   continue
      nsum    =  nsum + nn
700   continue
      call fault(xx,yy,dz)
      zgrid(index)  =  zgrid(index) + dz
      nsum    =  nsum - nn
      if(zgrid(index) .gt. zmax)  zmax  =  zgrid(index)
      if(zgrid(index) .lt. zmin)  zmin  =  zgrid(index)
800   continue
      return
      end
```

```
      subroutine hachur(x2,y2,x1,y1,delr,ik)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
           zgrid(65000),  xmax,          ymax,          zmax,
           istat(65000),  nx,            ny,            nz,
           x(16000),      nxgd,          nygd,          ncell,
           y(16000),      delgrd,        igrid,         icell,
           into,          zint,          zlevel,        level,
           nbold,         iplot,         dashl,         pcrang,
           levmin,        levmax,        ifit,          isw1,
           isw2,          isw3,          isw4,          isw5
      dimension           n(2,10),       xr(40),        yr(40),
         xb(2,200),       yb(2,200),     xe(2,200),     ye(2,200)
      data kabove,kbelow,ntot,iksv,n/24*0/
c
c  THIS SUBROUTINE IS DESIGNED TO PUT IN THE HACHURE MARKS TO
c  INDICATE THOSE CLOSED CONTOUR CURVES WHICH DENOTE THE FIRST
c  CHANGE OF SLOPE CAUSED BY A DEPRESSION.
c  THE TECHNIQUE USED FOR PLOTTING THESE HACHURES IS FIRST TO TEST
c  EACH CLOSED CONTOUR CURVE IN THE CALLING ROUTINE.  AT EACH
c  SPECIFIED DISTANCE INTERVAL ALONG THE CURVE THIS SUBROUTINE IS
c  CALLED.  THIS SUBROUTINE THEN TESTS TO SEE IF THIS IS A NEW
c  CONTOUR LEVEL.  IF IT IS, THEN THE SET OF HACHURE POINTS FOR THE
c  PREVIOUS LEVEL IS COMPLETE.  (THE LEVELS ARE ORDERED IN TERMS OF
c  INCREASING CONTOUR ELEVATION.)
c  THE PRINCIPAL VARIABLES ARE DEFINED AS FOLLOWS:
c    KBELOW  =  THE TOTAL NUMBER OF HACHURE SETS FOR THE LOWER
c               CONTOUR LEVEL.
c    KABOVE  =  THE TOTAL NUMBER OF HACHURE SETS FOR THE NEXT
c               HIGHER CONTOUR LEVEL.
c    N(2,I)  =  THE TOTAL NUMBER OF VALUES IN THE I-TH HACHURE
c               SET FOR THE LOWER CONTOUR LEVEL.
c    N(1,I)  =  THE TOTAL NUMBER OF VALUES IN THE I-TH HACHURE
c               SET FOR THE NEXT HIGHER CONTOUR LEVEL.
c    THE BEGINNING AND THE END POINTS OF EACH HACHURE ARE
c    STORED IN THE FOLLOWING ARRAYS:
c    XB(I,J) =  X-COORDINATE OF THE BEGINNING POINT.
c    YB(I,J) =  Y-COORDINATE OF THE BEGINNING POINT.
c    XE(I,J) =  X-COORDINATE OF THE ENDING POINT.
c    YE(I,J) =  Y-COORDINATE OF THE ENDING POINT.
c  IT SHOULD BE NOTED THAT THE INDEX,  I , TAKES ON A VALUE
c  OF TWO FOR THE LOWER CONTOUR LEVEL HACHURE SETS, AND A
c  VALUE OF ONE FOR THE NEXT HIGHER CONTOUR LEVEL HACHURE SETS.
c  THE INDEX,  J , REFERENCES EACH INDIVIDUAL HACHURE VALUE AS
```

```
c   IT HAS BEEN STORED IN THE HACHURE ARRAYS IN THE ORDER IN
c   WHICH IT WAS GENERATED FOR ITS GIVEN CONTOUR LEVEL.
c
c
c   THE VARIABLE, IK , IS A COUNTER WHICH SUMS THE NUMBER OF
c   COMPUTED HACHURE VALUES IN A GIVEN HACHURE SET.  THIS VALUE
c   IS INITIALIZED AT ZERO IN THE CALLING PROGRAM WHEN A NEW
c   CLOSED CONTOUR IS BEGUN.  THE VALUE OF  IK  IS STORED IN
c   THE VARIABLE, IKSV , SO THAT IT IS AVAILABLE FOR A FINAL
c   COUNT OF THE NUMBER OF VALUES IN A GIVEN HACHURE SET.  THIS IS
c   BECAUSE THE END OF A HACHURE SET IS NOT SENSED UNTIL COMPUT-
c   ATIONS ON A NEW CLOSED CONTOUR HAVE ALREADY BEGUN AND  IK  HAS
c   ALREADY BEEN SET TO ZERO.
c
c
c   IF THE HACHURE SETS FOR THE FIRST LEVEL ARE COMPLETE, GO TO
c   LOGIC TO TRANSFER DATA FROM UPPER LEVEL ARRAYS TO LOWER LEVEL
c   ARRAYS, SKIPPING THE PLOT LOGIC.  THEN RETURN TO BEGIN LOADING
c   HACHURE ARRAYS FOR THE NEXT CONTOUR LEVEL.
c
c
c   IF  IK  IS ZERO, A NEW CLOSED CONTOUR IS BEING STARTED.  IF THE
c   CONTOUR IS IN THE SAME CONTOUR LEVEL, INITIALIZE AND SET THE
c   THE RELEVANT INDICES.  IF THE CONTOUR LEVEL REMAINS THE SAME,
c   PROCEED TO THE LOGIC FOR ADDING HACHURE MARKS TO THE CONTOUR
c   CURVE.
c
          if(ik .gt. 0) go to 100
c
c   RESET COUNT VALUES FOR NEW CONTOUR CURVE.
c
1         kabove  =  kabove + 1
          ntot    =  ntot + iksv
          ilast   =  0
c
c   GIVEN A CONTOUR LINE SEGMENT FROM (X1,Y1) TO (X2,Y2) AND THE
c   DISTANCE,  DELR , FROM (X1,Y1), COMPUTE THE BEGINNING AND THE
c   ENDING POINTS FOR THE HACHURE ORTHOGONAL TO THAT POINT ON THE
c   CONTOUR.
100       ik      =  ik + 1
          iksv    =  ik
          n(1,kabove)  =  iksv
          d       =  sqrt((x2 - x1)**2 + (y2 - y1)**2)
          sinal   =  (y2 - y1)/d
          cosal   =  (x2 - x1)/d
          nik     =  ntot + ik
          xb(1,nik)  =  x1 + cosal*delr
          yb(1,nik)  =  y1 + sinal*delr
          xe(1,nik)  =  xb(1,nik) - .09*sinal
          ye(1,nik)  =  yb(1,nik) + .09*cosal
          return
c
c   FOR EACH INDEX HACHURE (USE THE FIRST XB,YB-COORDINATE OF THE
c   HACHURE SET) ON THE UPPER CONTOUR LEVEL, TEST TO SEE IF IT
c   IS CONTAINED IN A LOWER LEVEL HACHURE POLYGON.  IF IT IS,
c   CHANGE THE SIGN OF THE HACHURE POINT COUNT ASSOCIATED WITH
c   THAT SET OF HACHURE MARKS TO FLAG THAT IT LIES WITHIN THE
c   HACHURE POLYGON.
c   NEXT, TEST THE HACHURE SETS IN THE LOWER LEVEL.  IF A SET
c   CONTAINS AN INDEX POINT OF ANOTHER SET, AGAIN CHANGE THE SIGN
```

```
c  OF THE HACHURE COUNT ASSOCIATED WITH EACH SET WHICH CONTAINS
c  ANOTHER SET OF THE SAME LEVEL.
c
          entry hacnnd
          if(level .eq. levmax) ilast = 1
          if(n(2,1) .le. 0) go to 900
          n2       =  0
          do 600 i=1,kbelow
          n1       =  n2 + 1
          n2       =  n2 + n(2,i)
          do 300 j=n1,n2
          jj       =  j + 1 - n1
          xr(jj)   =  xb(2,j)
300       yr(jj)   =  yb(2,j)
          jj       =  jj + 1
          xr(jj)   =  xr(1)
          yr(jj)   =  yr(1)
          in       =  1
          do 400 k=1,kabove
          call polygn(xb(1,in),yb(1,in),xr,yr,jj,inreg)
          if(inreg .eq. 1) n(2,i) = - n(2,i)
400       in       =  in + abs(n(2,i))
          in       =  1
          do 500 k=1,kbelow
          if(k .eq. i) go to 500
          call polygn(xb(2,in),yb(2,in),xr,yr,jj,inreg)
          if(inreg .eq. 1) n(2,i) = - n(2,i)
500       in       =  in + aos(n(2,j))
600       continue
c
c  DELETE THOSE HACHURE SETS FOR THE LOWER LEVEL FOR WHICH THE
c  HACHURE COUNT VALUE IS NEGATIVE.
c
          n2       =  0
          nt       =  0
          do 700 i=1,kbelow
          if(n(2,i) .le. 0) go to 700
          n1       =  n2 + 1
          n2       =  n2 + n(2,i)
          kk       =  nt
          do 650 k=n1,n2
          kk       =  kk + 1
          xb(2,k)  =  xb(2,kk)
          yb(2,k)  =  yb(2,kk)
          xe(2,k)  =  xe(2,kk)
650       ye(2,k)  =  ye(2,kk)
700       nt       =  nt + abs(n(2,i))
c
c  IF THERE ARE NO CONTOURS LEFT TO PLOT IN THIS LEVEL, RESET
c  THE INDICATORS AND BEGIN FILLING IN THE HACHURE VALUES FOR
c  THE NEXT CONTOUR LEVEL.
c
          if(n2 .eq. 0) go to 900
c
c  PLOT THOSE HACHURE SETS BELONGING TO THE LOWER CONTOUR
c  LEVEL WHICH WERE NOT DISCARDED ABOVE.
c
          if(lplot .ge. 1)  call newpen(1)
790       do 800 i=1,n2
          call draws$move(xb(2,i),yb(2,i))
```

```
800         call draws(xe(2,i),ye(2,i))
c
c   THE FOLLOWING STATEMENTS TEST TO DETERMINE IF ALL CONTOURS
c   HAVE BEEN PLOTTED (I.E. ILAST = 1) AND IF THE LAST CONTOUR
c   LEVEL PLOTTED IS GREATER THAN THE FIRST LEVEL ABOVE THE LAST
c   CLOSED CONTOUR PLOTTED (I.E. LEVEL .GT. LEVSAV + 1).  IF
c   THESE TWO CRITERIA ARE SATISFIED, THEN THE LAST SET OF CLOSED
c   CONTOURS BELONGING TO THE HIGHEST LEVEL MUST, THEMSELVES,
c   BE TESTED TO ELIMINATE THOSE CLOSED CONTOURS WHICH CONTAIN
c   OTHER CLOSED CONTOURS OF THE SAME LEVEL.  THEN THE REMAINING
c   CLOSED CONTOUR SETS MUST HAVE THEIR HACHURE MARKS PLOTTED TO
c   INDICATE THAT THAT CLOSED CONTOUR DENOTES THE BEGINNING OF
c   A DEPRESSION.
c
c   AFTER THE LAST CASE IS TAKEN CARE OF, THE INDICES MUST ALL
c   BE SET TO ZERO TO ACCOMMODATE THE NEXT CONTOUR PLOT TO BE
c   COMPUTED DURING THE SAME PROGRAM EXECUTION.
c
            if(ilast .ne. 1) go to 900
            if(level .lt. levsav + 1) go to 890
            n2      =  0
            do 860 i=1,kabove
            n1      =  n2 + 1
            n2      =  n2 + n(1,i)
            do 840 j=n1,n2
            jj      =  j + 1 - n1
            xr(jj)  =  xb(1,j)
840         yr(jj)  =  yb(1,j)
            jj      =  jj + 1
            xr(jj)  =  xr(1)
            yr(jj)  =  yr(1)
            in      =  1
            do 850 k=1,kabove
            if(k .eq. i) go to 850
            call polygn(xb(1,in),yb(1,in),xr,yr,jj,inreg)
            if(inreg .eq. 1) n(1,i) = - n(1,i)
850         in      =  in + abs(n(1,j))
860         continue
            n2      =  0
            nt      =  0
            do 870 i=1,kabove
            if(n(1,i) .le. 0) go to 870
            n1      =  n2 + 1
            n2      =  n2 + n(1,i)
            kk      =  nt
            do 865 k=n1,n2
            kk      =  kk + 1
            xb(1,k) =  xb(1,kk)
            yb(1,k) =  yb(1,kk)
            xe(1,k) =  xe(1,kk)
865         ye(1,k) =  ye(1,kk)
870         nt      =  nt + abs(n(1,i))
            if(n2 .eq. 0) go to 890
            if(iplot .ge. 1)  call newpen(1)
            do 880 i=1,n2
            call draws$move(xb(1,i),yb(1,i))
880         call draws(xe(1,i),ye(1,i))
890         n(2,1)  =  0
            kabove  =  0
            kbelow  =  0
```

```
          ntot     =  0
          do 895 i=1,10
          do 895 j=1,2
895       n(j,i)   =  0
          return
c
c  EXCHANGE THE LEVEL VALUES AND THE COUNTERS TO PREPARE FOR THE
c  NEXT CONTOUR LEVEL.
c
900       kbelow   =  kabove
          kabove   =  0
          levsav   =  level
          ntot     =  0
          iksv     =  0
c
c  SHIFT THE UPPER LEVEL HACHURE VALUES INTO THE LOWER LEVEL
c  ARRAY AND PROCEED TO COMPUTE HACHURE VALUES FOR THE NEXT
c  HIGHER LEVEL.
c
          do 910 i=1,10
910       n(2,i)   =  n(1,i)
          do 920 i=1,nik
          xb(2,i)  =  xb(1,i)
          yb(2,i)  =  yb(1,i)
          xe(2,i)  =  xe(1,i)
920       ye(2,i)  =  ye(1,i)
          return
          end
```

```
      subroutine inside(is,inreg,xx,yy)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/     xb(20000,2),   yb(20000,2),   nch(2000,2),
                         noch(2),       kod(2000,2),   ipar(2),
                         icnd(2),       kap(15,2),     nptot(2)
      common /chain/     icf(100),      icl(100),      xint(101),
                         iptf(2000),    iptl(2000)
      common /corner/    xcorn(4),      ycorn(4)
      dimension          ysav(100),     dir(100),      ispar(100)
c
c     This subroutine tests the point (xx,yy) to determine if it is
c     inside of, or outside of, a given boundary array.  The array to
c     be tested is determined by the value of  is  (1 or 2).
c
c     The method for determining if the point is inside of, or outside
c     of, the boundary chains is to count the number of boundary line
c     segments lying below the point.  If the number of line segments
c     is odd, the point must lie inside of the boundary.  If the number
c     of line segments is even, the point must lie outside of the boundary.
c
c     If it is desired to include a region for plotting that lies outside
c     of a boundary, then the value of  ipar(is)  must be set to  1 .
c     A value of  0  for  ipar(is)  means that the plot region is inside
c     the boundary.
c
c
c     ic    counts the total number of intersections of the boundary
c           segments with the vertical ray through (xx,yy).
c
c     ibel  counts the number of line segments lying below the
c           point (xx,yy).
c
c     The following statements establish the boundary array indices
c     for the unsliced boundary chains.
c
      ibel      =  0
      i1        =  1
      i2        =  noch(is)
      jl        =  0
      do 10 i=1,i2
      jb        =  jl + 1
      jl        =  jl + iabs(nch(i,is))
      iptf(i)   =  jb
10    iptl(i)   =  isign(jl,nch(i,is))
```

```
          go to 50
c
c     This entry point is used to establish the boundary array
c     indices for the sliced boundary chains.
c
c     The slice interval containing the point (xx,yy) must first
c     be determined.  In order to avoid duplications at the slice
c     edges, all intervals except the first are defined to be
c     left-open and right-closed.  The first interval is closed
c     at both ends.
c
          entry inslic(is,inreg,xx,yy)
          ibel      =  0
          inreg     =  0
          ldex      =  1
          test      =  abs(xint(1) - xx)
          if(test .le. 0.)  go to 30
          do 20 l=1,100
          ldex      =  l
          test      =  (xint(l) - xx)*(xx - xint(l+1))
          if(test .gt. 0.)  go to 30
          test      =  abs(xint(l+1) - xx)
          if(test .le. 0.)  go to 30
20        continue
          return
30        continue
          i1        =  icf(ldex)
          i2        =  icl(ldex)
          if(i1 .gt. i2)  go to 600
50        ic        =  0
          nc        =  nach(is)
          if(nc .eq. 0)  go to 600
c
c     Scan through all of the line segments of all of the chains in the
c     is-th array to determine all intersections with the vertical ray
c     passing through the point (xx,yy).  Store the y-coordinates of the
c     intersection points in the  ysav  array.  Also store the direction
c     of the midpoint of the line segment with respect to the vertical
c     ray in the  dir  array so that the value of +1 indicates that it
c     lies to the right of the ray, the value of -1 indicates that it
c     lies to the left of the ray, and the value 0 indicates that the
c     line segment is coincident with the vertical ray.  If the line
c     segment is coincident with the vertical ray, store both y-values
c     of the endpoints in the  ysav  array.
c
          do 200 i=i1,i2
          jb        =  iptf(i)
          x1        =  xb(jb,is)
          y1        =  yb(jb,is)
          jb        =  jb + 1
          jl        =  iabs(iptl(i))
          do 200 j=jb,jl
          x2        =  xb(j,is)
          y2        =  yb(j,is)
c
c     If the x-coordinate,  xx , of the point does not lie between
c     x1  and  x2 , then go on to test the next line segment.
c
          if((x1-xx)*(xx-x2) .lt. 0.)  go to 100
```

```
c
c
c     If the line segment is vertical and it is coincident with the
c     vertical ray, store the endpoints in the  ysav  array and set
c     tne value of  dir  to 0.
c
          test1     =  abs(xx - x1)
          test2     =  abs(xx - x2)
          if(test1 + test2 .gt. 0.)  go to 90
          ic        =  ic + 1
          ysav(ic)  =  y1
          dir(ic)   =  0.
          ic        =  ic + 1
          ysav(ic)  =  y2
          dir(ic)   =  0.
          go to 100
c     Compute the y-value on the line segment at the x-value of  xx
c     and store it in the  ysav  array.  Also store the direction
c     of the midpoint of the line segment with respect to the
c     vertical ray through the x-coordinate  xx , in the  dir
c     array, so that the value of +1 indicates that it lies to
c     the right of the ray, the value -1 indicates that it
c     lies to the left, and the value 0 indicates that the line
c     segment is coincident with the vertical ray.  If the line
c     segment is coincident with the vertical ray, store the
c     y-values of the endpoints in the  ysav  array.
c
c
90        ic        =  ic + 1
          ysav(ic)  =  y1 + (y2 - y1)*(xx - x1)/(x2 - x1)
          dir(ic)   =  sign(1.,.5*(x1 + x2) - xx)
100       x1        =  x2
          y1        =  y2
200       continue
c
c     If there are no boundary line segments intersecting the vertical
c     ray through  xx , determine the value of  inreg  from  ipar  and
c     return.
c
          if(ic .eq. 0)  go to 600
c
c     If there is only one value stored in the  ysav-array, do not
c     sort the array.  Test the value to see if it lies below  yy ,
c     and increment  ibel  accordingly.
c
          if(ic .ge. 2)  go to 300
          if(ysav(1) .le. yy)  ibel = 1
          go to 600
c
c     If there is more than one value stored in the  ysav-array,
c     sort this array in ascending order and tnen delete the
c     duplicate values before counting the number of points
c     which lie below  yy .
c
300       idx       =  1
          call dsort(ysav,dir,ic)
          tol       =  .0000001*abs(ycorn(3)-ycorn(1))
          ispar(1)  =  1
          do 450 i=2,ic
          if(abs(ysav(i)-ysav(i-1)) .gt. tol)  go to 440
```

```
c
c     A duplicate intersection point has been encountered.
c     Discard the duplicate point, but update the  dir  array
c     so that if one of the segments was vertical, the value
c     of  dir  will indicate to which side of the vertical
c     ray that the nonvertical segment lies.
c
          dir(idx)=  dir(i-1) + dir(i)
c
c     If the following test is satisfied, both line segments
c     producing the duplicate intersection point lie on the
c     same side of the vertical ray.  Do not count them when
c     determining the parity.
c
          if(abs(dir(idx)) .gt. 1.5)  ispar(idx) = 0
c
c     If one of the duplicate intersection points resulted from
c     a vertical line segment, set the value of  ispar  to  -1 .
c     The absolute value of  ispar  will be used later to deter-
c     mine the parity of the count of the number of line
c     segments lying below the value  yy .
c
          if(abs(abs(dir(idx)) - 1.0) .lt. tol)  ispar(idx) = -1
          if(idx .le. 1)  go to 450
c
c     If the previous duplicate value that was loaded came from
c     a vertex with a vertical line segment and this duplicate
c     intersection value also has a vertical line segment, then
c     test to see if the nonvertical line segments creating the
c     vertices on the vertical ray intersect the ray from opposite
c     sides.  If so, set the value of  ispar  to 2 so that the
c     parity does not change at the second intersection.  If
c     the two line segments enter from the same side,  keep the
c     parity odd.
c
          itest     =  ispar(idx-1) + ispar(idx)
          test      =  dir(idx-1)*dir(idx)
          if(itest .eq. -2  .and.  test .gt. 0)  ispar(idx) = 1
          if(itest .eq. -2  .and.  test .lt. 0)  ispar(idx) = 2
          go to 450
440       idx       =  idx + 1
          ysav(idx) =  ysav(i)
          dir(idx)  =  dir(i)
          ispar(idx)=  1
450       continue
c
c     Count the number of values stored in  ysav  which lie
c     below  yy .
c
          do 500 i=1,idx
500       if(ysav(i) .le. yy)  ibel = ibel + iabs(ispar(i))
c
c     If the number of line segments plus the parity indicator is
c     even, then  inreg       = 0 .  This means that the point (xx,yy) lies
c     outside the the desired bounded region.  If the parity is odd,
c     then the point lies inside the desired region.
c
600       inreg     =  mod(ibel + ipar(is),2)
          return
          end
```

```
      subroutine line2(x1,y1,x2,y2,xr,yr,iflag)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /grapnc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          isw1,
     8     isw2,          isw3,          isw4,          isw5
      double precision    xq,            yq
      dimension      x1(2),     y1(2),     x2(2),     y2(2)
c
c*****************************************************************
c  THIS SUBROUTINE COMPUTES THE POINT OF INTERSECTION BETWEEN
c  TWO LINES GIVEN IN POINT FORM.
c  IFLAG  =  1   MEANS THAT THE LINES INTERSECT BETWEEN DATA POINTS
c  IFLAG  =  0   MEANS THAT THERE IS NO INTERSECTION BETWEEN THE
c                 DATA POINTS
c*****************************************************************
c
      tol       =  .1e-5*sqrt((xmax - xmin)**2 + (ymax - ymin)**2)
      dx1    =  x1(2) - x1(1)
      dy1    =  y1(2) - y1(1)
      dx2    =  x2(2) - x2(1)
      dy2    =  y2(2) - y2(1)
      if(abs(dy1*dx2 - dy2*dx1) .lt. .1e-10) go to 400
      if(abs(dx1) .gt. .1e-10) go to 100
      xr     =  x1(1)
      yq     =  y2(1) + dy2/dx2*(xr - x2(1))
      yr     =  round(yq)
      go to 300
100   if(abs(dx2) .gt. .1e-10) go to 200
      xr     =  x2(1)
      yq     =  y1(1) + dy1/dx1*(xr - x1(1))
      yr     =  round(yq)
      go to 300
200   xq     =  (y1(1)-y2(1))*dx1*dx2 +x2(1)*dy2*dx1 -x1(1)*dy1*dx2
      xq     =  xq/(dx1*dy2 - dy1*dx2)
      xr     =  round(xq)
      yq     =  y2(1) + dy2/dx2*(xr - x2(1))
      yr     =  round(yq)
      if(abs(dy1) .le. .1e-10)  yr = y1(1)
```

```
      if(abs(dy2) .le. -.1e-10)  yr = y2(1)
300   if((x1(1)-xr)*(xr-x1(2)) .lt. -tol*abs(x1(2)-x1(1)))  go to 400
      if((x2(1)-xr)*(xr-x2(2)) .lt. -tol*abs(x2(2)-x2(1)))  go to 400
      if((y1(1)-yr)*(yr-y1(2)) .lt. -tol*abs(y1(2)-y1(1)))  go to 400
      if((y2(1)-yr)*(yr-y2(2)) .lt. -tol*abs(y2(2)-y2(1)))  go to 400
      iflag      =  1
      if(sqrt((x1(1)-xr)**2 + (y1(1)-yr)**2) .gt. tol)  go to 310
      xr         =  x1(1)
      yr         =  y1(1)
      return
310   if(sqrt((x1(2)-xr)**2 + (y1(2)-yr)**2) .gt. tol)  go to 320
      xr         =  x1(2)
      yr         =  y1(2)
      return
320   if(sqrt((x2(1)-xr)**2 + (y2(1)-yr)**2) .gt. tol)  go to 330
      xr         =  x2(1)
      yr         =  y2(1)
      return
330   if(sqrt((x2(2)-xr)**2 + (y2(2)-yr)**2) .gt. tol)  return
      xr         =  x2(2)
      yr         =  y2(2)
      return
400   iflag    =  0
      return
      end
```

```
      subroutine logic(is)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /switch/ isobs,icrt,idum3,itek,isdum(6)
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          kset,
     8     ndig,          ktour,         ismth,         icnt
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
     1                    nocn(2),       kod(2000,2),   ipar(2),
     2                    icnd(2),       kap(15,2),     nptot(2)
      common /files/      filnm1,        filnm2,        filnm3,
     1                    filgrd,        filobs
      character*8         filnm1,        filnm2,        filnm3,
     1                    filgrd,        filobs
      dimension           x1(2),         y1(2),         ysav(50),
     1                    x2(2),         y2(2)
      data iblank/" "/
      isn       =  3 - is
c
c     The name of boundary chain no. 1 is  filnm1
c     The name of boundary chain no. 2 is  filnm2
c     To create a new boundary set, enter a:
c       1 , if you want all of no. 1 and/or no. 2
c       2 , if you want only what is in both no. 1 and no. 2
c       3 , if you want only what is in no. 1, but not in no. 2
c       4 , if you want only what is in no. 2, but not in no. 1
c
c     The name of the resultant boundary file is stored in filenm3
c
c     The value of  iopt  determines which set of boundary chains
c     will be selected.
c
100       call newpag
          call bell
          call anmode
          write(6,1006)
1001      format(" TO CREATE A NEW BOUNDARY SET, ENTER A:")
          write(6,1001)
1002      format("   1 , IF YOU WANT ALL THAT IS IN ",a8,
```

```
                " AND/OR ",a8)
          write(6,1002) filnm1,filnm2
1003      format("   2 , IF YOU WANT ONLY WHAT IS IN BOTH ",a8,
                " AND ",a8)
          write(6,1003) filnm1,filnm2
1004      format("   3 , IF YOU WANT ONLY WHAT IS IN ",a8,
                " BUT NOT IN ",a8)
          write(6,1004) filnm1,filnm2
1009      format("   4 , IF YOU WANT ONLY WHAT IS IN ",a8,
                " BUT NOT IN ",a8)
          write(6,1009) filnm2,filnm1
          write(6,1006)
          call prompt("ENTER BOUNDARY COMBINATION CODE: ",33)
          read(5,1008) iopt
1008      format(i1)
          write(6,1006)
          call prompt("IF YOU WISH TO SAVE THIS FILE, ENTER NAME: ",43)
          read(5,1005) filnm3
1005      format(a8)
c
c     Initialize the plot window and then plot all of the
c     chains from the concatenated file.  When the plot is
c     finished, the bell will sound so that the user may
c     continue by pressing the carriage return.
c
          call crtbeg(xmin,xmax,ymin,ymax)
          noc       =  nocn(is)
          jb        =  1
          do 102 i=1,noc
          jl        =  jb + ncn(i,is) - 1
          call movea(xb(jb,is),yb(jb,is))
          jbp       =  jb + 1
          do 101 j=jbp,jl
101       call dasha(xb(j,is),yb(j,is),1)
102       jb        =  jl + 1
          call bell
          call anmode
1006      format(2a4)
c
c     Load the descriptors for the specified logical
c     combination into the  kap  array.
c
          kap(3,isn)=  "  fr"
          kap(4,isn)=  "om  "
          if(iopt .ne. 1)  go to 120
          kap(8,isn)=  " "
          kap(9,isn)=  " uni"
          kap(10,isn)= "on  "
          kap(11,isn)=  " "
          go to 160
120       kap(12,isn)=  " "
          if(iopt .ne. 3)  go to 130
          kap(12,isn)= "not "
          go to 150
130       kap(5,isn)=  " "
          if(iopt .ne. 4)  go to 140
          kap(5,isn)=  "not "
          go to 150
140       if(iopt .ne. 2)  go to 100
150       kap(8,isn)=  "  in"
```

```
          kap(9,isn)=  "ters"
          kap(10,isn)= "ecti"
          kap(11,isn)= "on  "
c
c     Begin the process of selecting those chains from the
c     concatenated boundary (is-th) array which meet the
c     requirements for the logical (iopt) selection.
c
c          noc     indexes and counts the number of chains
c                  being selected.
c
c          np      indexes and counts the total number of points
c                  being selected.
c
c          nc      is the total number of chains to be processed
c                  from the concatenated (is-th) array.
c
160       noc       =  0
          np        =  0
          nc        =  noch(is)
          call czaxis(1)
c
c     Scan through the chains in the concatenated (is-th) array.
c     Select those chains whose  kod  values correspond to
c     criteria specified by the value of  iopt  and store the
c     selected chains in the isn-th array.
c
          jb        =  1
          do 300 i=1,nc
          jl        =  jb + nch(i,is) - 1
          if(iopt .ne. 1)  go to 210
c
c     A union between region no. 1 and region no. 2 has been specified.
c     Select those boundary chains from region no. 1 which lie outside
c     region no. 2 (kod = 3) and those boundary chains from region no. 2
c     which lie outside region no. 1 (kod = 4).  In addition, select
c     those coincident boundary subchains such that region no. 1 and
c     region no. 2 both lie on the same side of the subchain (kod = 5).
c
          if(kod(i,is) .eq. 3  .or.  kod(i,is) .eq. 4  .or.
             kod(i,is) .eq. 5)  go to 250
          go to 300
210       if(iopt .ne. 2)  go to 220
c
c     An intersection between region no. 1 and region no. 2 has been
c     specified.  Select those boundary chains from region no. 1 which
c     lie inside region no. 2 (kod = 1) and those boundary chains from
c     region no. 2 which lie inside region no. 1 (kod = 2).  In addition
c     select those coincident boundary subchains such that region no. 1
c     and region no. 2 both lie on the same side of the subchain
c     (kod = 5).
c
          if(kod(i,is) .eq. 1  .or.  kod(i,is) .eq. 2  .or.
             kod(i,is) .eq. 5)  go to 250
          go to 300
220       if(iopt .ne. 3)  go to 230
c
c     An intersection between region no. 1 and everything outside of
c     region no. 2 has been specified.  Select those boundary chains
c     from region no. 1 which lie outside region no. 2 (kod = 3) and
```

```
c      those boundary chains from region no. 2 which lie inside
c      region no. 1 (kod = 2).  In addition, select those coincident
c      chains such that region no. 1 and region no. 2 lie on opposite
c      sides of the subchain (kod = 6).
c
           if(kod(i,is) .eq. 2  .or.  kod(i,is) .eq. 3  .or.
              kod(i,is) .eq. 6)  go to 250
           go to 300
230        if(iopt .ne. 4)  go to 9999
c
c      An intersection between region no. 2  and everything outside of
c      region no. 1 has been specified.  Select those boundary chains
c      from region no. 1 which lie inside region no. 2 (kod = 1) and
c      those boundary chains from region no. 2 which lie outside
c      region no. 1 (kod = 4).  In addition, select those coincident
c      subchains such that region no. 1 and region no.2 lie on opposite
c      sides of the boundary subchain (kod = 6).
c
c      If there is not a  kod  value of 1, 2, 3, 4, 5, or 6 associated
c      with every chain in the is-th array, then the array is not a
c      legitimate concatenated array and an error exit to statement  9999
c      will be taken.
c
           if(kod(i,is) .eq. 1  .or.  kod(i,is) .eq. 4 .or.
              kod(i,is) .eq. 6)  go to 250
           go to 300
c
c      A chain meeting the criteria specified by the value of  iopt
c      has been selected.  Load this chain into the isn-th array
c      and also load the  nch  value associated with this
c      chain into the isn-th array of  nch  and the  kod
c      value into the corresponding position of the isn-th array
c      of  kod .
c
c      Also, the selected chain will be overplotted on the boundary
c      display in bold outline with a solid line denoting that the
c      chain bounds region no. 1 and a dasned line denoting that the
c      chain bounds region no. 2.
c
250        noc        =  noc + 1
           call movea(xb(jb,is),yb(jb,is))
           nch(noc,isn)  =  nch(i,is)
           kod(noc,isn)  =  kod(i,is)
           itest      =  mod(kod(i,is),2)
           do 260 j=jb,jl
           np         =  np + 1
           xb(np,isn)=  xb(j,is)
           yb(np,isn)=  yb(j,is)
           if(j .eq. jb)  go to 260
           if(itest .eq. 1)  call drawa(xb(np,isn),yb(np,isn))
           if(itest .eq. 0)  call dasha(xb(np,isn),yb(np,isn),3)
260        continue
300        jb         =  jl + 1
           nptot(isn)=  np
           noch(isn) =  noc
c
c      The bell will sound and the cursor will appear on the screen.
c      Locate the cursor at a position in the boundary plot to select
c      an area to be included as part of the region of interest.  To
c      enter this point, press the letter "i" on the terminal keyboard.
```

```
c      Depending upon the number of boundaries crossed, all bounded
c      regions will be either included or excluded as part of the
c      region of interest, based on their relationship to the region
c      located by the cursor.  The value of  ipar  is set to establish
c      the parity of the regions for inclusion or exclusion.
c
310         call bell
            call movabs(1,3100/(3-itek)**2)
            call anmode
c$$         read(5,1006)
c$$         write(6,1007)
c$$1007     format(" PLACE THE CURSOR INSIDE THE"/
c$$                " REGION YOU WISH TO INCLUDE"/
c$$                " AND DEPRESS THE 'i' KEY.")
c$$         call stty("-modes","^erkl,ctl_char")
c$$         call vcursr(itest,xin,yin)
c$$         call stty("-modes","erkl,^ctl_char")
c$$         if(itest .ne. 105)  go to 310
c$$         ipar(isn) =  0
c$$         call inside(isn,inout,xin,yin)
c$$         ipar(isn) =  1 - inout
c$$         call shade(isn)
c$$         call anmode
            icrt      =  1
            call complt(isn)
            icrt      =  0
c$$         read(5,1006) itest
            return
9999        write(6,1111)
1111        format(/" YOU DID NOT SPECIFY A VALID BOUNDARY",
                   " COMBINATION CODE."/" THE BOUNDARY",
                   " COMBINATION PROCEDURE IS TERMINATED."/)
            return
            end
```

```
      subroutine logo
c
c           This subroutine was written and programmed by
c                           A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      call twindo(1,976,1,720)
      call czaxis(0)
      call movabs(100,600)
      call drwabs(272,700)
      call drwabs(688,700)
      call drwabs(860,600)
      call drwabs(688,500)
      call drwabs(272,500)
      call drwabs(100,600)
      call czaxis(1)
      call drwabs(164,600)
      call movabs(272,700)
      call drwabs(272,665)
      call movabs(688,700)
      call drwabs(688,665)
      call movabs(860,600)
      call drwabs(796,600)
      call movabs(688,500)
      call drwabs(688,535)
      call movabs(272,500)
      call drwabs(272,535)
      call movabs(184,600)
      call drwabs(272,650)
      call drwabs(688,650)
      call drwabs(776,600)
      call drwabs(688,550)
      call drwabs(272,550)
      call drwabs(184,600)
      call movabs(320,576)
      call chrsiz(1)
      call anmode
      write(6,1001)
      call movabs(318,400)
      call czaxis(0)
      call chrsiz(3)
      call anmode
      write(6,1002)
      call movabs(301,325)
      call czaxis(1)
      call anmode
      write(6,1003)
      call movabs(380,225)
```

```
      call czaxis(0)
      call chrsiz(4)
      call anmode
      write(6,1004)
      call chrsiz(3)
      call movabs(352,150)
      call anmode
      write(6,1005)
      call movabs(256,85)
      call anmode
      write(6,1006)
      call movabs(300,20)
      call anmode
      write(6,1007)
      return
1001  format(" GARNET")
1002  format(" created by")
1003  format(" A. C. OLSON")
1004  format(" for the")
1005  format(" NATIONAL")
1006  format(" COAL RESOURCES")
1007  format(" DATA SYSTEM")
      end
```

```
      subroutine messag
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      dimension          lpt(1500)
      common /surfac/    xob(1500),     yob(1500),
     1    zt(1500),      zob(1500),     residu(1500),
     2    m,             n,             np,
     3    perd,          resum,         ntc,
     4    npc,           msq,           msq2,
     5    msq3,          msq4,          sig
      common /graphc/    xmin,          ymin,          zmin,
     1    zgrid(65000),  xmax,          ymax,          zmax,
     2    istat(65000),  nx,            ny,            nz,
     3    x(16000),      nxgd,          nygd,          ncell,
     4    y(16000),      delgrd,        igrid,         icell,
     5    into,          zint,          zlevel,        level,
     6    nbold,         lplot,         dashl,         pcrang,
     7    levmin,        levmax,        ifit,          isw1,
     8    isw2,          isw3,          isw4,          isw5
      common /baud/ ibaud
      write(6,1003)
1003  format(/" THE DATA VALUES WHICH EXCEED 3 TIMES THE ROOT MEAN"
     1" SQUARE ERROR ARE:"//7x,"XOB",9x,"YOB",9x,"ZOB",7x,"ERROR"/
     25x,"-------",5x,"-------",5x,"-------",5x,"-----")
      err      =  3.*sig
      do 300 i=1,np
      lpt(i)   =  0
      if(abs(residu(i)) .le. err) go to 300
      lpt(i)   =  1
      write(6,1004) xob(i),yob(i),zob(i),residu(i)
300   continue
1005  format(/" THE DATA VALUES WHICH EXCEED 2 TIMES THE ROOT MEAN"
     1" SQUARE ERROR ARE:"//7x,"XOB",9x,"YOB",9x,"ZOB",7x,"ERROR"/
     25x,"-------",5x,"-------",5x,"-------",5x,"-----")
      write(6,1005)
      err      =  2.*sig
      do 400 i=1,np
      if(lpt(i) .ne. 0) go to 400
      if(abs(residu(i)) .le. err) go to 400
      write(6,1004) xob(i),yob(i),zob(i),residu(i)
400   continue
1004  format(2(5x,f7.3),5x,f7.1,3x,f7.1)
      call hdcopy
      call initt(ibaud/10)
      write(6,1002)
```

```
1002    format(//" THESE DATA VALUES WILL ALSO BE DISPLAYED ON THE"/
        " FOLLOWING CONTOUR PLOT FOR EDITING."/
        " TO DELETE A VALUE FROM THE DATA FILE, PLACE THE CURSOR ON"/
        " THE POINT AND PRESS THE ""D"" KEY."/
        " TO CHANGE A DATA VALUE, PLACE THE CURSOR ON THE POINT AND"/
        " PRESS THE ""C"" KEY.  THE DATA VALUE MUST BE THEN ENTERED"/
        " FOLLOWING THE PROMPT FOR ""VALUE."""/
        " TO PROCEED TO THE NEXT STEP IN THE EDITING PROCESS, PRESS"/
        " THE ""N"" KEY.")
        return
        end
```

```
      subroutine obsin(c)
c
c           This subroutine was written and programmed by
c                          A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      character*8 filnm1,filnm2,filnm3,filgrd,filobs,filnam
      dimension       c(69)
      common /switch/       isobs,isdum(9)
      common /files/ filnm1,filnm2,filnm3,filgrd,filobs
      common /surfac/      xob(1500 ),      yob(1500 ),
           zt(1500 ),      zob(1500 ),      residu(1500 ),
           m,              n,               np,
           perd,           resum,           ntc,
           npc,            msq,             msq2,
           msq3,           msq4,            sig
      common /graphc/      xmin,            ymin,            zmin,
           zgrid(65000),   xmax,            ymax,            zmax,
           istat(65000),   nx,              ny,              nz,
           x(16000),       nxgd,            nygd,            ncell,
           y(16000),       delgrd,          igrid,           icell,
           into,           zint,            zlevel,          level,
           nbold,          lplot,           dashl,           pcrang,
           levmin,         levmax,          ifit,            isw1,
           isw2,           ktour,           ismth,           icnt
      common /corner/      xcorn(4),        ycorn(4)
50    call prompt("NAME OF OBSERVED POINT FILE: ",29)
      read(5,1003)filnam
1003  format(a8)
      if(filnam.eq." ".and.filobs.eq." ") go to 50
      isobs=1
      if(filnam.eq.filobs) return
      if(filnam.eq." ") return
      if(filnam.ne."none") go to 200
      isobs=0
      return
200   call assoc(22,filnam,"si  ")
      read(22,1004,err=100) np,m,n,xmin,xmax,ymin,ymax,zmin,zmax,sig,
           xcorn,ycorn
1004  format(3i5/7f10.4/8f10.4)
      read(22,1005,err=100) (xob(i),yob(i),zob(i),i=1,np)
      read(22,1005,err=100) (residu(i),i=1,np)
      read(22,1005,err=100) c
1005  format(6e12.5)
      rewind 22
      call assoc$closer(22)
      filobs=filnam
      return
```

```
100       call assoc$closer(22)
          go to 50
          entry obsout(c)
          write(6,1002)
1002      format(" A-NEW OBSERVED POINT FILE IS BEING CREATED, PROVIDE")
          call prompt("NAME OF OBSERVED POINT FILE: ",29)
          read(5,1003)filnam
          call assoc(21,filnam,"so  ")
          if(abs(xmax - xmin) .gt. 0.  .and.  abs(ymax - ymin) .gt. 0.)  go to 150
          xmin      =  amin1(xcorn(1),xcorn(2),xcorn(3),xcorn(4))
          xmax      =  amax1(xcorn(1),xcorn(2),xcorn(3),xcorn(4))
          ymin      =  amin1(ycorn(1),ycorn(2),ycorn(3),ycorn(4))
          ymax      =  amax1(ycorn(1),ycorn(2),ycorn(3),ycorn(4))
150       write(21,1004) np,m,n,xmin,xmax,ymin,ymax,zmin,zmax,sig,
                    xcorn,ycorn
          write(21,1005)(xob(i),yob(i),zob(i),i=1,np)
          write(21,1005)(residu(i),i=1,np)
          write(21,1005) c
          call assoc$closer(21)
          filobs=filnam
          return
          end
```

```
      subroutine outlin(radrel,nocirp)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /surfac/    xob(1500),      yob(1500),
     1     zt(1500),     zob(1500),      residu(1500),
     2     m,            n,              np,
     3     perd,         resum,          ntc,
     4     npc,          msq,            msq2,
     5     msq3,         msq4,           sig
      common /graphc/    xmin,           ymin,           zmin,
     1     zgrid(65000), xmax,           ymax,           zmax,
     2     istat(65000), nx,             ny,             nz,
     3     x(16000),     nxgd,           nygd,           ncell,
     4     y(16000),     delgrd,         igrid,          icell,
     5     into,         zint,           zlevel,         level,
     6     nbold,        lplot,          dashl,          pcrang,
     7     levmin,       levmax,         ifit,           isw1,
     8     isw2,         isw3,           isw4,           isw5
      common /bound/ xb(20000,2),   yb(20000,2),   nch(2000,2),
     1               noch(2),       kod(2000,2),   ipar(2),
     2               icnd(2),       kap(15,2),     nptot(2)
c
c  THIS SUBROUTINE COMPUTES THE RESOURCE BOUNDARIES.  GIVEN A
c  SET OF OBSERVATION POINTS (XOB(I),YOB(I)), A RADIUS FOR
c  RELIABILITY OF MEASUREMENT, RADREL, AND REGIONS BOUNDED
c  EXTERNALLY OR INTERNALLY BY POLYGONS, THIS SUBROUTINE COMPUTES
c  THE BOUNDARY OUTLINES WHICH LIE IN ACCEPTABLE POLYGONALLY
c  DEFINED REGIONS BUT DO NOT LIE WITHIN THE RADIUS OF RELIABILITY
c  OF ANY OF THE OTHER OBSERVATION POINTS.
c  INOUT  =  1  INDICATES THAT A BOUNDARY POINT LIES OUTSIDE ALL
c               OTHER RELIABILITY CIRCLES, BUT INSIDE ALL
c               ACCEPTABLE POLYGONALLY DEFINED REGIONS.
c  INOUT  =  0  INDICATES THAT A BOUNDARY POINT LIES INSIDE AT
c               LEAST ONE OTHER RELIABILITY CIRCLE, OR ELSE
c               OUTSIDE THE ACCEPTABLE POLYGONALLY DEFINED
c               REGIONS.
c  INSAV  =  THE  INOUT  VALUE FOR THE PRECEDING BOUNDARY POINT.
c
      inreg     =  1
      iflag     =  0
      is        =  2
      nc        =  nocirp + 1
      xnc       =  nocirp
      xrad      =  6.2831853/xnc
      do 900 i=1,np
```

```
          do 900 k=1,nc
          jj       =  np + 1
          dqk      =  k - 1
          dqk      =  xrad*dqk
          xc       =  xob(i) + radrel*cos(dqk)
          yc       =  yob(i) + radrel*sin(dqk)
          if((xmin - xc)*(xc - xmax) .lt. 0.) go to 200
          if((ymin - yc)*(yc - ymax) .lt. 0.) go to 200
c
c  THE FOLLOWING STATEMENTS TEST FIRST OF ALL TO SEE IF THE
c  BOUNDARY POINT ON A CIRCLE, (XC,YC), LIES WITHIN AN ACCEPTABLE
c  POLYGONALLY DEFINED REGION.  IF SO, THE POINT IS THEN TESTED
c  TO DETERMINE IF IT LIES OUTSIDE ALL OTHER RELIABILITY CIRCLES.
c  IF A CIRCLE CROSSES OVER THE BOUNDARY OF ANOTHER CIRCLE OR A
c  POLYGON, FIND THE INTERSECTION, (XI,YI).
c
          distmx   =  2.*radrel
          do 100 j=1,np
          if(i .eq. j) go to 100
          dist     =  sqrt((xc - xob(j))**2 + (yc - yob(j))**2)
          if(dist .ge. distmx) go to 100
          jj       =  j
          distmx   =  dist
100       continue
          if(distmx .le. radrel) go to 200
          jj       =  np + 1
          if(kap(15,1) .ne. "no") call inside$inslic(is,inreg,xc,yc)
          if(inreg .eq. 0) go to 200
c
c  THE BOUNDARY POINT, (XC,YC), LIES OUTSIDE ALL OTHER RELIABILITY
c  CIRCLES, BUT WITHIN ALL ACCEPTABLE POLYGONALLY DEFINED REGIONS.
c  IF THE PRECEDING POINT IS OF THE SAME STATUS, DRAW A LINE
c  BETWEEN THE TWO POINTS.  OTHERWISE, PROCEED TO LOGIC TO FIND
c  THE INTERSECTION OF ONE OF THE POLYGONAL OR CIRCULAR BOUNDARIES
c  WITH THE LINE SEGMENT BETWEEN THE TWO POINTS.
c
          inout    =  1
          if(k .eq. 1) go to 800
          xi       =  xsav
          yi       =  ysav
          if(insav .ne. 1) go to 300
          if(k .eq. 2) call draws$move(xsav,ysav)
          call draws(xc,yc)
          go to 800
c
c  THE BOUNDARY POINT, (XC,YC), LIES INSIDE AT LEAST ONE OF THE
c  RELIABILITY CIRCLES, OR OUTSIDE THE ACCEPTABLE POLYGONALLY
c  DEFINED REGIONS.  IF THE PRECEDING POINT IS OF THE SAME STATUS,
c  NO LINE IS DRAWN.  THE POINTS ARE SHIFTED, AND A NEW POINT IS
c  CALCULATED AND TESTED.  IF THE STATUS IS DIFFERENT, PROCEED TO
c  LOGIC TO FIND THE INTERSECTION OF THE POLYGONAL OR CIRCULAR
c  BOUNDARY WITH THE LINE SEGMENT BETWEEN THE TWO POINTS.
c
200       inout    =  0
          xi       =  xc
          yi       =  yc
          if(insav .eq. 0  .or.  k .eq. 1) go to 800
c
c  THIS IS THE LOGIC FOR DETERMINING THE INTERSECTION.
c  ==== == === ===== === =========== === =============
```

```
c  FIRST TEST TO SEE IF CIRCLE INTERSECTS WITH POLYGON.
c
300       if(kap(15,1) .ne. "no") call bndint(xsav,ysav,xc,yc,xb,yb,iflag,is)
          if(iflag .eq. 0) go to 400
          xi       = xb
          yi       = yb
c
c  IF THE CIRCLE IN QUESTION INTERSECTS WITH A POLYGON, ALSO
c  TEST TO SEE IF IT INTERSECTS WITH ANY OTHER CIRCLES WITHIN
c  THE POLYGONAL REGION.  IF SO, THE POLYGONAL INTERSECTION
c  POINT WILL BE REPLACED WITH THE INTERSECTION WITH THE OTHER
c  CIRCLE.
c
400       if(jj .gt. np  .and.  jjsav .gt. np)  go to 600
c
c  DETERMINE THE POINT OF INTERSECTION WITH THE OTHER CIRCLE.
c
          jk       = jj
          if(insav .lt. inout)  jk = jjsav
          call cirint(xsav,ysav,xc,yc,xob(jk),yob(jk),radrel,xs1,ys1,
                                  xs2,ys2,int)
          if(int .eq. 0) go to 600
          if((xsav - xs1)*(xs1 - xc) .lt. 0.) go to 500
          if((ysav - ys1)*(ys1 - yc) .lt. 0.) go to 500
          xi       = xs1
          yi       = ys1
          go to 600
500       if((xsav - xs2)*(xs2 - xc) .lt. 0.) go to 600
          if((ysav - ys2)*(ys2 - yc) .lt. 0.) go to 600
          xi       = xs2
          yi       = ys2
c
c  TEST TO DETERMINE IF A RELIABILITY CIRCLE INTERSECTS THE MAP
c  EDGE.  IF SO, DETERMINE THE INTERSECTION POINT AND USE IT TO
c  TERMINATE THE RELIABILITY BOUNDARY.
c
600       if(xi .le. xmax) go to 610
          yi       = ysav + (yc - ysav)*(xmax - xsav)/(xc - xsav)
          xi       = xmax
          go to 620
610       if(xi .ge. xmin) go to 620
          yi       = ysav + (yc - ysav)*(xmin - xsav)/(xc - xsav)
          xi       = xmin
620       if(yi .le. ymax) go to 630
          xi       = xsav + (xc - xsav)*(ymax - ysav)/(yc - ysav)
          yi       = ymax
          go to 640
630       if(yi .ge. ymin) go to 640
          xi       = xsav + (xc - xsav)*(ymin - ysav)/(yc - ysav)
          yi       = ymin
c
c  IF THE LINE SEGMENT FROM THE BOUNDARY OF A CIRCLE MOVES OUT OF
c  A RESTRICTED REGION, PLOT THE LINE SEGMENT TO THE INTERSECTION.
c
640       if(insav .lt. inout) go to 700
          call draws$move(xsav,ysav)
          call draws(xi,yi)
          go to 800
c
c  IF THE LINE SEGMENT FROM THE BOUNDARY OF A CIRCLE MOVES INTO
```

```
c  A RESTRICTED REGION, SKIP TO THE INTERSECTION AND THEN PLOT THE
c  THE REMAINDER OF THE LINE SEGMENT.
c
700       call draws$move(xi,yi)
          call draws(xc,yc)
c
c  SHIFT THE POINTS AND THE STATUS VALUES AND PROCEED TO PROCESS
c  THE NEXT POINT.
c
300       insav    =  inout
          jisav    =  jj
          xsav     =  xc
          ysav     =  yc
900       continue
          return
          end
```

```
      subroutine path(klose)
c
c          This subroutine was written and programmed by
c                         A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /index/ nx1,          nx2,          ny1,          ny2,
                     nxgm,         nygm,         xorig,        yorig
      common /graphc/     xmin,         ymin,         zmin,
           zgrid(65000),  xmax,         ymax,         zmax,
           istat(65000),  nx,           ny,           nz,
           x(16000),      nxgd,         nygd,         ncell,
           y(16000),      delgrd,       igrid,        icell,
           into,          zint,         zlevel,       level,
           nbold,         lplot,        dashl,        pcrang,
           levmin,        levmax,       ifit,         kset,
           isw2,          ktour,        ismth,        icnt
      dimension           lh(4),        z1(4),        z2(4)
      xi       =  x(1)
      yi       =  y(1)
      test     =  delgrd
      icount   =  1
      if(ismtn .eq. 1) call smooth(xi,yi,1)
100   qix      =  igrid/ny - nx1 + 1
      qiy      =  mod(igrid,ny) - ny1
      igdny    =  igrid + ny
      z1(1)    =  zgrid(igrid)
      z1(2)    =  zgrid(igrid+1)
      z1(3)    =  zgrid(igdny+1)
      z1(4)    =  zgrid(igdny)
      do  10 i=1,4
      ii       =  mod(i-into+3,4) + 1
      z2(ii)   =  z1(i)
      lh(ii)   =  0
10    if(z2(ii) .gt. zlevel) lh(ii) = 1
      if(lh(1) + lh(2) + lh(3) + lh(4) .eq. 0)  go to 40
c
c  TEST TO SEE IF CONTOUR LINE EXITS FROM OPPOSITE SIDE OF THE
c  GRID CELL.
c
16    if(lh(1) .ne. lh(2)  .and.  lh(3) .ne. lh(4))  go to 40
      if(lh(1) .ne. lh(2)  .or.  lh(3) .ne. lh(4)) go to 20
      iout     =  mod(into+1,4) + 1
      go to 50
c
c  TEST TO SEE IF CONTOUR LINE EXITS FROM THE RIGHT EDGE OF THE
c  GRID CELL.
c
```

```
20          if(lh(1) .ne. lh(2)) go to 30
21          iout    =  mod(into+2,4) + 1
            go to 50
c
c  TEST TO SEE IF CONTOUR LINE EXITS FROM THE LEFT EDGE OF THE
c  GRID CELL.
c
30          if(ln(3) .ne. lh(4)) go to 40
31          iout    =  mod(into,4) + 1
            go to 50
c
c  IT IS POSSIBLE FOR THE CONTOUR LINE TO EXIT THE GRID CELL
c  FROM EITHER THE RIGHT OR THE LEFT SIDE.  DETERMINE THE SLOPE
c  OF THE CONTOUR LINE AT THE ENTRY POINT BY MEANS OF A BILINEAR
c  FIT TO THE FOUR GRID POINTS.  THE SLOPE INFORMATION IS THEN
c  USED TO DETERMINE THE SIDE OF THE GRID CELL THROUGH WHICH THE
c  CONTOUR LINE WILL EXIT.
c
40          xx      =  xi - xorig - qix*delgrd
            yy      =  yi - yorig - qiy*delgrd
            if(into .eq. 4) go to 43
            if(into .ne. 1) go to 41
            xx      =  delgrd - yy
            go to 43
41          if(into .ne. 2) go to 42
            xx      =  delgrd - xx
            go to 43
42          xx      =  yy
43          a       =  z2(1)
            b       =  (z2(4) - z2(1))/delgrd
            c       =  (z2(2) - z2(1))/delgrd
            d       =  (z2(1) - z2(2) + z2(3) - z2(4))/delgrd**2
            if(abs(c+d*xx) .le. 0.) go to 44
            slope   =  -b/(c + d*xx)
            if(slope .gt. 0.) go to 21
            if(slope .lt. 0.) go to 31
44          if(xx .gt. .5*delgrd) go to 21
            go to 31
50          go to (210,220,230,240),iout
210         zp1     =  z1(1) - zlevel
            zp2     =  zlevel - z1(2)
            qinc    =  .5
            if(abs(zp1+zp2) .gt. 0.)  qinc = zp1/(zp1 + zp2)
            xi      =  xorig + delgrd*qix
            yi      =  yorig + delgrd*(qiy + qinc)
            if(icell .le. nygd*nx1)  go to 400
            go to 300
220         zp1     =  z1(2) - zlevel
            zp2     =  zlevel - z1(3)
            qinc    =  .5
            if(abs(zp1+zp2) .gt. 0.)  qinc = zp1/(zp1 + zp2)
            xi      =  xorig + delgrd*(qix + qinc)
            yi      =  yorig + delgrd*(qiy + 1.)
            if(mod(icell-1,nygd) .ge. ny2 - 2)  go to 400
            go to 300
230         zp1     =  z1(3) - zlevel
            zp2     =  zlevel - z1(4)
            qinc    =  .5
            if(abs(zp1+zp2) .gt. 0.)  qinc = zp2/(zp1 + zp2)
            xi      =  xorig + delgrd*(qix + 1.)
```

```
        yi      =  yorig + delgrd*(qiy + qinc)
        if(icell .gt. nygd*(nxgm - 1)) go to 400
        go to 300
240     zp1     =  z1(4) - zlevel
        zp2     =  zlevel - z1(1)
        qinc    =  .5
        if(abs(zp1+zp2) .gt. 0.)  qinc = zp2/(zp1 + zp2)
        xi      =  xorig + delgrd*(qix + qinc)
        yi      =  yorig + delgrd*qiy
        if(mod(icell,nygd) .eq. ny1) go to 400
300     istat(icell)  =  istat(icell) + iout - 1
        test    =  sqrt((x(1) - xi)**2 + (y(1) - yi)**2)
        if(test .le. .001*delgrd) go to 400
        if(iout .ne. 1) go to 310
        into    =  3
        igrid   =  igrid - ny
        icell   =  icell - nygd
        go to 340
310     if(iout .ne. 2) go to 320
        into    =  4
        igrid   =  igrid + 1
        icell   =  icell + 1
        go to 340
320     if(iout .ne. 3) go to 330
        into    =  1
        igrid   =  igrid + ny
        icell   =  icell + nygd
        go to 340
330     into    =  2
        igrid   =  igrid - 1
        icell   =  icell - 1
340     istat(icell)  =  4*into
        icount  =  icount + 1
        x(icount)  =  xi
        y(icount)  =  yi
        go to 100
400     icount  =  icount + 1
        if(klose .eq. 1 .and. icount .le. 6)  return
        x(icount)  =  xi
        y(icount)  =  yi
        if(test .le. .001*delgrd) call cclock(area,x,y,icount)
        if(icount .le. 3) go to 430
        if(ismth .ne. 1) go to 430
        do 410 i=1,icount
        ix      =  16000 - icount + i
        x(ix)   =  x(i)
410     y(ix)   =  y(i)
        do 420 i=1,icount
        ix      =  16000 - icount + i
        ic      =  i
        if(ic .ne. icount) go to 420
        ic      =  -1
        if(test .le. .001*delgrd) ic = 0
420     call smooth(x(ix),y(ix),ic)
500     call curve(icnt)
        inout   =  -1
        if(ktour .eq. 1  .or.  kset .lt. 0) call conin$conout(klose,inout)
        return
430     icnt    =  icount
        go to 500
```

```
end
```

```
      subroutine planar(gfact)
c
c           This subroutine was written and programmed by
c                           A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /surfac/     xob(1500),      yob(1500),
           zt(1500),      zob(1500),      residu(1500),
           m,             n,              np,
           perd,          resum,          ntc,
           npc,           msq,            msq2,
           msq3,          msq4,           sig
      common /graphc/     xmin,           ymin,           zmin,
           zgrid(65000),  xmax,           ymax,           zmax,
           istat(65000),  nx,             ny,             nz,
           x(16000),      nxgd,           nygd,           ncell,
           y(16000),      delgrd,         igrid,          icell,
           into,          zint,           zlevel,         level,
           nbold,         lplot,          dashl,          pcrang,
           levmin,        levmax,         ifit,           isw1,
           isw2,          ktour,          ismth,          icnt
      dimension           sum(3,3),       b(3)
      dimension           ul(3,3),        soln(3)
      dimension           indx(24),       dist(24)
      zmin     =  1.e20
      zmax     = -1.e20
      maxpts   =  16
      if(np .ge. 4  .and.  np .le. 24)  maxpts = np
      dmax     =  sqrt((xmax - xmin)**2 + (ymax - ymin)**2)
      do 200 j=1,nx
      xx       =  j - 1
      xx       =  xx*delgrd + xmin
      do 200 i=1,ny
      yy       =  i - 1
      yy       =  yy*delgrd + ymin
      index    =  (j-1)*ny + i
      sumw     =  0.
      sumwx    =  0.
      sumwy    =  0.
      sumwz    =  0.
      sumwxx   =  0.
      sumwxy   =  0.
      sumwyy   =  0.
      sumwzx   =  0.
      sumwzy   =  0.
      distmx   =  dmax
      do 50 k=1,np
      dista    =  sqrt((xob(k) - xx)**2 + (yob(k) - yy)**2)
```

```
      if(k .gt. maxpts  .and.  dista .ge. distmx) go to 50
      kdex    =  k
      kmax    =  k
      if(kmax .gt. maxpts) kmax  =  maxpts
      do 40 l=1,kmax
      lmin    =  l + 1
      if(l .eq. kmax) go to 35
      if(dista .gt. dist(l)) go to 40
      do 30 m=lmin,kmax
      mdex    =  kmax + lmin - m
      dist(mdex)  =  dist(mdex-1)
30    indx(mdex)  =  indx(mdex-1)
35    dist(lmin-1)  =  dista
      indx(lmin-1)  =  kdex
      go to 45
40    continue
45    if(k .gt. maxpts) distmx  = dist(maxpts)
50    continue
      do 100 k=1,maxpts
      l       =  indx(k)
      q       =  24.*dist(k)/distmx
      wt      =  720./(720. + q*(720. + q*(360. + q*(120. + q*(30.
                    + q*(6. + q))))))
      dumz    =  wt*zob(l)
      dumx    =  wt*(xob(l) - xx)
      dumy    =  wt*(yob(l) - yy)
      sumw    =  sumw + wt
      sumwx   =  sumwx + dumx
      sumwy   =  sumwy + dumy
      sumwz   =  sumwz + dumz
      sumwxx  =  sumwxx + dumx*(xob(l) - xx)
      sumwxy  =  sumwxy + dumx*(yob(l) - yy)
      sumwyy  =  sumwyy + dumy*(yob(l) - yy)
      sumwzx  =  sumwzx + dumx*zob(l)
      sumwzy  =  sumwzy + dumy*zob(l)
100   continue
      sumw    =  sumw + gfact
      sumwz   =  sumwz + gfact*zgrid(index)
      sum(1,1)  =  sumw
      sum(1,2)  =  sumwx
      sum(2,1)  =  sumwx
      sum(1,3)  =  sumwy
      sum(3,1)  =  sumwy
      sum(2,2)  =  sumwxx
      sum(2,3)  =  sumwxy
      sum(3,2)  =  sumwxy
      sum(3,3)  =  sumwyy
      b(1)      =  sumwz
      b(2)      =  sumwzx
      b(3)      =  sumwzy
      call linear(sum,soln,b,ul,3,3)
      zgrid(index)  =  soln(1)
      if(zgrid(index) .gt. zmax)  zmax = zgrid(index)
      if(zgrid(index) .lt. zmin)  zmin = zgrid(index)
200   continue
      return
      end
```

```
      subroutine pltchn(is)
c
c           This subroutine was written and programmed by
c                          A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
           zgrid(65000),  xmax,          ymax,          zmax,
           istat(65000),  nx,            ny,            nz,
           x(16000),      nxgd,          nygd,          ncell,
           y(16000),      delgrd,        igrid,         icell,
           into,          zint,          zlevel,        level,
           nbold,         lplot,         dashl,         pcrang,
           levmin,        levmax,        ifit,          kset,
           ncig,          isw3,          ismth,         isw5
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
                          noch(2),       kod(20000,2),  ipar(2),
                          icnd(2),       kap(15,2),     nptot(2)
c
c     This subroutine is designed to scan through the  is-th  chain
c     array and plot each chain on the CRT display.
c
c           noch(is)     gives the total number of chains in the  is-th
c                        array.
c
c           nch(i,is)    gives the number of points in the  i-th chain
c                        of the  is-th  array.
c
c           jb           indexes the beginning point of the chain.
c
c           jl           indexes the last point of the chain.
c
      jb        =  1
      nc        =  noch(is)
      do 200 i=1,nc
      if(nch(i,is) .lt. 0)  return
      jl        =  jb + nch(i,is) - 1
      xx1       =  xb(jb,is)
      yy1       =  yb(jb,is)
      jbp       =  jb + 1
      call draws$move(xx1,yy1)
      do 150 j=jbp,jl
      xx2       =  xb(j,is)
      yy2       =  yb(j,is)
      t1x       =  (xmin - xx1)*(xx1 - xmax)
      t2x       =  (xmin - xx2)*(xx2 - xmax)
      t1y       =  (ymin - yy1)*(yy1 - ymax)
c
```

```
c     If both points of the boundary line segment are within the
c     rectangular frame, plot the entire line segment.
c
          t2y     .  =  (ymin - yy2)*(yy2 - ymax)
          if(t1x .ge. 0. .and. t2x .ge. 0. .and.
             t1y .ge. 0. .and. t2y .ge. 0.)  go to 50
          t3x        =  (xx1 - xmin)*(xmin - xx2)
          t4x        =  (xx1 - xmax)*(xmax - xx2)
          t3y        =  (yy1 - ymin)*(ymin - yy2)
          t4y        =  (yy1 - ymax)*(ymax - yy2)
c
c     If the boundary line segment does not intersect the vertical
c     lines through  xmin  or  xmax , nor the horizontal lines
c     through  ymin  or  ymax , skip to the next line segment in
c     the boundary chain.
c
          if(t3x .lt. 0. .and. t4x .lt. 0. .and.
             t3y .lt. 0. .and. t4y .lt. 0.)  go to 100
c
c     If one of the endpoints of the boundary line segment
c     lies within the rectangular frame, proceed to the
c     logic for determining the intersection with the frame
c     and plot the portion of the line segment that is inside
c     the frame.
c
          if((t1x .ge. 0. .and. t1y .ge. 0.) .or.
             (t2x .ge. 0. .and. t2y .ge. 0.))  go to 10
          xxm        =  .5*(xx1 + xx2)
          yym        =  .5*(yy1 + yy2)
          t4x        =  (xmin - xxm)*(xxm - xmax)
          t4y        =  (ymin - yym)*(yym - ymax)
c
c     If both end points of the boundary line segment lie
c     outside of the rectangular frame, and a test of the
c     midpoint of the line segment shows that it, too, is
c     outside the frame, then proceed to logic for processing
c     the next line segment.
c
          if(t4x .lt. 0. .or. t4y .lt. 0.)  go to 100
c
c     A part of the boundary line segment lies within the
c     rectangular frame.  Start with the first point of the
c     line segment (itest = 0) and find the closest frame
c     intersection.  If the point lies within the frame, an
c     intersection will not be computed and the value of
c     (xx,yy) will be that of the first endpoint.  Then
c     process the second endpoint (itest = 1) and find the
c     closest frame intersection.  If this point lies within
c     the frame, an intersection is not computed and the
c     value of (xx,yy) remains that of the second endpoint.
c     A move to the first value of (xx,yy) and a draw to the
c     second value of (xx,yy) draws that portion of the
c     line segment lying within the frame.
c
c
10        xx         =  xx1
          yy         =  yy1
          itest      =  0
20        if((xmin - xx)*(xx - xmax) .ge. 0.)  go to 30
          if((xx1 - xmin)*(xmin - xx2) .gt. 0.)  xx = xmin
```

```
      if((xx1 - xmax)*(xmax - xx2) .gt. 0.)  xx = xmax
      if(abs(xx2-xx1) .gt. 0.)  yy = yy1 + (yy2-yy1)*(xx-xx1)/(xx2-xx1)
30    if((ymin - yy)*(yy - ymax) .ge. 0.)  go to 40
      if((yy1 - ymin)*(ymin - yy2) .gt. 0.)  yy = ymin
      if((yy1 - ymax)*(ymax - yy2) .gt. 0.)  yy = ymax
      if(abs(yy2-yy1) .gt. 0.)  xx = xx1 + (xx2-xx1)*(yy-yy1)/(yy2-yy1)
40    if(itest .eq. 0)  call draws$move(xx,yy)
      if(itest .eq. 1)  call draws(xx,yy)
      xx        =  xx2
      yy        =  yy2
      itest     =  itest + 1
      if(itest .le. 1)  go to 20
      go to 100
50    call draws(xx2,yy2)
100   xx1       =  xx2
      yy1       =  yy2
150   continue
200   jb        =  jl + 1
      return
      end
```

```
      suoroutine points
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /surfac/     xob(1500),      yob(1500),
           zt(1500),      zob(1500),      residu(1500),
           m,             n,              np,
           perd,          resum,          ntc,
           npc,           msq,            msq2,
           msq3,          msq4,           sig
      common /graphc/     xmin,           ymin,           zmin,
           zgrid(65000),  xmax,           ymax,           zmax,
           istat(65000),  nx,             ny,             nz,
           x(16000),      nxgd,           nygd,           ncell,
           y(16000),      delgrd,         igrid,          icell,
           into,          zint,           zlevel,         level,
           nbold,         lplot,          dashl,          pcrang,
           levmin,        levmax,         ifit,           isw1,
           isw2,          isw3,           ismth,          icnt
      common /factor/fact
      xrng     =  xmax - xmin
      yrng     =  ymax - ymin
      do 100 i=1,np
      xtest    =  (xmin - xob(i))*(xob(i) - xmax)
      ytest    =  (ymin - yob(i))*(yob(i) - ymax)
      if(xtest .le. 0.  .or.  ytest .le. 0.) go to 100
      call cross(xob(i),yob(i))
      yy       =  yob(i) -(.0042*yrng + .05)
      xx       =  xob(i) - .042
      if(lplot .ge. 1) call calbeg$calnum(xx,yy,.006*yrng*fact,zob(i),0.,-1)
      xx       =  xob(i) - .05
      yy       =  yob(i) - .02*yrng
      if(lplot .le. 1) call crtnum(xx,yy,zob(i),6,1)
100   continue
      return
      end
```

```
      subroutine qdgrid(gfact)
c
c           This subroutine was written and programmed by
c                              A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /surfac/     xob(1500),     yob(1500),
     1     zt(1500),      zob(1500),     residu(1500),
     2     m,             n,             np,
     3     perd,          resum,         ntc,
     4     npc,           msq,           msq2,
     5     msq3,          msq4,          sig
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     npold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          isw1,
     8     isw2,          ktour,         ismth,         icnt
      dimension           sum(6,6),      b(6)
      dimension           ul(6,6),       soln(6)
      dimension           indx(24),      dist(24)
      zmin     =  1.e20
      zmax     = -1.e20
      maxpts   =  16
      if(np .ge. 16  .and.  np .le. 24)  maxpts = np
      dmax     =  sqrt((xmax - xmin)**2 + (ymax - ymin)**2)
      do 200 j=1,nx
      xx       =  j - 1
      xx       =  xx*delgrd + xmin
      do 200 i=1,ny
      yy       =  i - 1
      yy       =  yy*delgrd + ymin
      index    =  (j-1)*ny + i
      sw       =  0.
      sx       =  0.
      sy       =  0.
      sx2      =  0.
      sxy      =  0.
      sy2      =  0.
      sx3      =  0.
      sx2y     =  0.
      sxy2     =  0.
      sy3      =  0.
      sx4      =  0.
      sx3y     =  0.
```

```
      sx2y2   =  0.
      sxy3    =  0.
      sy4     =  0.
      sumz    =  0.
      sumzx   =  0.
      sumzy   =  0.
      sumzx2  =  0.
      sumzxy  =  0.
      sumzy2  =  0.
      distmx  =  amax
      do  50 k=1,np
      dista   =  sqrt((xob(k) - xx)**2 + (yob(k) - yy)**2)
      if(k .gt. maxpts  .and.  dista .ge. distmx)  go to 50
      kdex    =  k
      kmax    =  k
      if(kmax .gt. maxpts)  kmax  =  maxpts
      do  40 l=1,kmax
      lmin    =  l + 1
      if(l .eq. kmax)  go to 35
      if(dista .gt. dist(l))  go to 40
      do  30 m=lmin,kmax
      mdex    =  kmax + lmin - m
      dist(mdex)  =  dist(mdex-1)
30    indx(mdex)  =  indx(mdex-1)
35    dist(lmin-1)  =  dista
      indx(lmin-1)  =  kdex
      go to 45
40    continue
45    if(k .gt. maxpts)  distmx  =  dist(maxpts)
50    continue
      do 100 k=1,maxpts
      l       =  indx(k)
      q       =  30.0*dist(k)/distmx
      wt      =  720./(720. + q*(720. + q*(360. + q*(120. + q*(30.
                 + q*(6. + q))))))
      dumz    =  wt*zob(l)
      dumx    =  wt*(xob(l) - xx)
      dumy    =  wt*(yob(l) - yy)
      dumx2   =  dumx*(xob(l) - xx)
      dumxy   =  dumx*(yob(l) - yy)
      dumy2   =  dumy*(yob(l) - yy)
      dumx3   =  dumx2*(xob(l) - xx)
      dumy3   =  dumy2*(yob(l) - yy)
      sw      =  sw + wt
      sx      =  sx + dumx
      sy      =  sy + dumy
      sx2     =  sx2 + dumx2
      sxy     =  sxy + dumxy
      sy2     =  sy2 + dumy2
      sx3     =  sx3 + dumx3
      sx2y    =  sx2y + dumx2*(yob(l) - yy)
      sxy2    =  sxy2 + dumy2*(xob(l) - xx)
      sy3     =  sy3 + dumy3
      sx4     =  sx4 + dumx3*(xob(l) - xx)
      sx3y    =  sx3y + dumx3*(yob(l) - yy)
      sx2y2   =  sx2y2 + dumx2*dumy2/wt
      sxy3    =  sxy3 + dumy3*(xob(l) - xx)
      sy4     =  sy4 + dumy3*(yob(l) - yy)
      sumz    =  sumz + dumz
      sumzx   =  sumzx + dumx*zob(l)
```

```
      sumzy   =  sumzy + dumy*zob(l)
      sumzx2  =  sumzx2 + dumx2*zob(l)
      sumzxy  =  sumzxy + dumxy*zob(l)
      sumzy2  =  sumzy2 + dumy2*zob(l)
100   continue
      sw      =  sw + gfact
      sumz    =  sumz + gfact*zgrid(index)
      sum(1,1)  =  sw
      sum(1,2)  =  sx
      sum(2,1)  =  sx
      sum(1,3)  =  sy
      sum(3,1)  =  sy
      sum(2,2)  =  sx2
      sum(1,4)  =  sx2
      sum(4,1)  =  sx2
      sum(2,3)  =  sxy
      sum(3,2)  =  sxy
      sum(1,5)  =  sxy
      sum(5,1)  =  sxy
      sum(2,4)  =  sx3
      sum(4,2)  =  sx3
      sum(3,3)  =  sy2
      sum(1,6)  =  sy2
      sum(6,1)  =  sy2
      sum(2,5)  =  sx2y
      sum(5,2)  =  sx2y
      sum(3,4)  =  sx2y
      sum(4,3)  =  sx2y
      sum(2,6)  =  sxy2
      sum(6,2)  =  sxy2
      sum(3,5)  =  sxy2
      sum(5,3)  =  sxy2
      sum(4,4)  =  sx4
      sum(3,6)  =  sy3
      sum(6,3)  =  sy3
      sum(4,5)  =  sx3y
      sum(5,4)  =  sx3y
      sum(4,6)  =  sx2y2
      sum(6,4)  =  sx2y2
      sum(5,5)  =  sx2y2
      sum(5,6)  =  sxy3
      sum(6,5)  =  sxy3
      sum(6,6)  =  sy4
      b(1)    =  sumz
      b(2)    =  sumzx
      b(3)    =  sumzy
      b(4)    =  sumzx2
      b(5)    =  sumzxy
      b(6)    =  sumzy2
      call linear(sum,soln,b,ul,6,6)
      zgrid(index)  =  soln(1)
      if(zgrid(index) .gt. zmax)  zmax  =  zgrid(index)
      if(zgrid(index) .lt. zmin)  zmin  =  zgrid(index)
200   continue
      return
      end
```

```
      subroutine regstr(itest,*)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
     1    zgrid(65000),  xmax,          ymax,          zmax,
     2    istat(65000),  nx,            ny,            nz,
     3    x(16000),      nxgd,          nygd,          ncell,
     4    y(16000),      delgrd,        igrid,         icell,
     5    into,          zint,          zlevel,        level,
     6    nbold,         lplot,         dashl,         pcrang,
     7    levmin,        levmax,        ifit,          isw1,
     8    isw2,          isw3,          isw4,          isw5
      common /param/      nx1,      ny1,      nz1,
     1                    nx2,      ny2,      nz2,
     2    xmin1,    xmax1,    ymin1,    ymax1,    zmin1,    zmax1,
     3    xmin2,    xmax2,    ymin2,    ymax2,    zmin2,    zmax2,
     4    delgd1,   xcorn1(4),          ycorn1(4),
     5    delgd2,   xcorn2(4),          ycorn2(4)
      common /corner/     xcorn(4),          ycorn(4)
      itest    =  0
      nz       =  (nz1 + nz2)/2
      if(nx1 .eq. nx2) go to 301
      itest    =  1
      write(6,1301) nx1,nx2
1301  format(/" THE FIRST GRID SET HAS",i4," POINTS IN THE X-DIRECTION"
     1    ", THE SECOND HAS",i4,".")
301   if(ny1 .eq. ny2) go to 302
      itest    =  1
      write(6,1302) ny1,ny2
1302  format(/" THE FIRST GRID SET HAS",i4," POINTS IN THE Y-DIRECTION"
     1    ", THE SECOND HAS",i4,".")
302   if(sqrt((xmin1-xmin2)**2 + (ymin1-ymin2)**2)/
     1   sqrt((xmin1-xmax1)**2 + (ymin1-ymax1)**2) .le. .00001)
     2   go to 303
      write(6,1303)
1303  format(/" WARNING: THE ORIGINS FOR EACH GRIDDED DATA SET DO"
     1    " NOT REGISTER.")
303   if(abs((xmax1-xmin1)/(xmax2-xmin2) - 1.) .le. .0002) go to 304
      itest    =  1
      xrang1   =  xmax1 - xmin1
      xrang2   =  xmax2 - xmin2
      write(6,1304) xrang1,xrang2
1304  format(/" THE HORIZONTAL DIMENSION OF THE FIRST GRID SET IS",f7.3,
     1    " AND OF THE SECOND IS",f7.3,".")
304   if(abs((ymax1-ymin1)/(ymax2-ymin2) - 1.) .le. .0002) go to 305
```

```
      itest   =  1
      yrang1  =  ymax1 - ymin1
      yrang2  =  ymax2 - ymin2
      write(6,1305) yrang1,yrang2
1305  format(/" THE VERTICAL DIMENSION OF THE FIRST GRID SET IS",f7.3,
     1    " AND OF THE SECOND IS",f7.3,".")
305   if(abs(delgd1 - delgd2) .le. .00001) go to 306
      itest   =  1
      write(6,1305) delgd1,delgd2
1306  format(/" THE GRID INTERVAL FOR THE FIRST GRID SET IS",f7.3," AND"
     1    " FOR THE SECOND, ",f7.3,".")
306   ik      =  0
      do 307 i=1,4
      if(abs(xcorn1(i) - xcorn2(i)) .gt. .01) ik  =  1
      if(abs(ycorn1(i) - ycorn2(i)) .gt. .01) ik  =  1
      xcorn(i)  =  xcorn1(i)
307   ycorn(i)  =  ycorn1(i)
      if(itest .eq. 1)  return1
      if(ik .ne. 1) go to 308
      write(6,1309)
1309  format(/" WARNING: THE CORNERS OF THE MAPS MAY NOT NECESSARILY"
     1    " COINCIDE.  THE CORNER "/" POINTS FOR THE FIRST "
     2    "GRIDDED SURFACE WILL BE USED FOR THE PLOTTED DATA."/)
308   return
      end
```

```
      real function round(x)
c
c           This subroutine was written and programmed by
c                              A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      double precision     x,          y
      dimension            k(2),       c(2)
      equivalence          (c,y,k),    (rnd,irnd)
      data ione/o0000000000001/
      round    =  0.
      if(x .eq. 0.d0) return
      y        =  x
      irnd     =  k(1)
      if(k(2) .lt. 0)  irnd = irnd + ione
      round    =  sign(rnd,c(1))
      return
      end
```

```
      subroutine scrin(is,file1,file2,*)
c
c          This subroutine was programmed by
c                        R. J. Smith
c          to the specifications of A. C. Olson for use in
c          the GARNET interactive graphics system.  GARNET
c          was developed to perform resource mapping and
c          resource estimations for the   NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/ xb(20000,2),yb(20000,2),nch(2000,2),noch(2),
     kod(2000,2),ipar(2),icnd(2),kap(15,2),nptot(2)
      common /files/ filnm1,filnm2,filnm3,filgrd,filobs
      character*8 filnm1,filnm2,filnm3,filgrd,filobs,
     file1,file2
      character iscrth*23
      data iscrth/"GARNET.BOUNDARY.SCRATCH"/
      call io("attach","file42","vfile_",iscrth)
      call io("open","file42","si")
      read(42,200,end=180,err=180) (kap(i,is),i=1,15),nptot(is),noch(is),
     icnd(is),ipar(is)
      encode(filnm1,100) kap(6,is),kap(7,is)
100   format(2a4)
      encode(filnm2,100) kap(13,is),kap(14,is)
      if(filnm1.ne.file1) go to 110
      if(filnm2.ne.file2) go to 180
      go to 120
110   if(filnm1.ne.file2) go to 180
      if(filnm2.ne.file1) go to 180
120   read(42,210,end=180,err=180) (nch(i,is),i=1,noch(is))
      read(42,220,end=180,err=180) (kod(i,is),i=1,noch(is))
      read(42,230,end=180,err=180) (xb(i,is),yb(i,is),i=1,nptot(is))
      call closer(42)
      return 1
      entry scrout(is)
      call io("attach","file41","vfile_",iscrth)
      call io("open","file41","so")
      write(41,200) (kap(i,is),i=1,15),nptot(is),noch(is),
     icnd(is),ipar(is)
      write(41,210) (nch(i,is),i=1,noch(is))
      write(41,220) (kod(i,is),i=1,noch(is))
      write(41,230) (xb(i,is),yb(i,is),i=1,nptot(is))
      call closer(41)
      return
180   call closer(42)
      return
200   format(15a4,4i5)
210   format(16i5)
220   format(80i1)
230   format(8f10.4)
      end
```

```
      subroutine shade(is)
c
c           This subroutine was written and programmed by
c                           A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
           zgrid(65000),  xmax,          ymax,          zmax,
           istat(65000),  nx,            ny,            nz,
           x(16000),      nxgd,          nygd,          ncell,
           y(16000),      delgrd,        igrid,         icell,
           into,          zint,          zlevel,        level,
           nbold,         lplot,         dashl,         pcrang,
           levmin,        levmax,        ifit,          kset,
           ndig,          ktour,         ismth,         icnt
      common /bound/      xb(20000,2),   yb(20000,2),   nch(2000,2),
                          noch(2),       kod(2000,2),   ipar(2),
                          icnd(2),       kap(15,2),     nptot(2)
      dimension           x1(2),         y1(2),         ysav(50),
                          x2(2),         y2(2),         dir(50),
                          ispar(50)
c
c     The following set of statements is used to plot a vertical
c     crosshatch pattern to denote that region of the logically
c     combined boundary that has been selected for inclusion as
c     the region of interest.
c
c     The boundary map is divided into one hundred and one equally
c     spaced vertical rays.  Each ray is then tested against the
c     boundary chains contained in the is-th chain array to
c     determine the y-coordinates of each, if any, intersection
c     with a line segment from the boundary chain.
c
c     After all of the intersection points have been determined,
c     the y-coordinates are sorted in ascending order.  Then each
c     point is taken in order, and the parity of the rank
c     plus the value of  ipar  is used to determine if that portion
c     of the ray is to be plotted as a solid line or if it is to
c     be left blank.
c
          call czaxis(0)
          tol       =  10.*(ymax - ymin)
          noc       =  noch(is)
          delta     =  .01*(xmax - xmin)
          y2(1)     =  ymin - tol
          y2(2)     =  ymax + tol
          tol       =  5.e-8*tol
          do 600 k=1,101
```

```
            qkm         =  k - 1
            xx          =  xmin + qkm*delta
            x2(1)       =  xx
            x2(2)       =  xx
            call movea(xx,ymin)
            jb          =  1
            icnt        =  0
            do 400 i=1,noc
            jl          =  jb + iabs(nch(i,is)) - 1
            x1(1)       =  xb(jb,is)
            y1(1)       =  yb(jb,is)
c
c       Scan through the chains in the is-th array to determine the
c       intersection.  Store the y-coordinates of the intersection
c       points in the  ysav  array.  Also store the direction of the
c       midpoint of the line segment with respect to the vertical
c       shading ray in the  dir  array so that the value of +1
c       indicates that it lies to the right of the ray, the value
c       -1 indicates that it lies to the left, and the value 0
c       indicates that the line segment is coincident with the
c       vertical ray.  If the line segment is coincident with the
c       vertical ray, store both y-values of the endpoints in the
c       ysav  array.
c
            jbp         =  jb + 1
            isbar(1)    =  1
            do 390 j=jbp,jl
            x1(2)       =  xb(j,is)
            y1(2)       =  yb(j,is)
c
c       Test to see if the line segment is vertical and if the
c       xx -value is on the vertical ray passing throuth the line
c       segment.  If so, load the y-value for both of the endpoints
c       into the  ysav -array, set the direction value,  dir , to
c       zero, and move on to test the next line segment.
c
            test1       =  abs(xx - x1(1))
            test2       =  abs(xx - x1(2))
            if(test1 + test2 .gt. 0.)  go to 370
            icnt        =  icnt + 1
            ysav(icnt)=  y1(1)
            dir(icnt)   =  0.
            icnt        =  icnt + 1
            ysav(icnt)=  y1(2)
            dir(icnt)   =  0.
            go to 380
370         call line2(x1,y1,x2,y2,xr,yr,iflag)
            if(iflag .eq. 0)  go to 380
            icnt        =  icnt + 1
            dir(icnt)   =  sign(1.,.5*(x1(1) + x1(2)) - xx)
            ysav(icnt)=  yr
380         x1(1)       =  x1(2)
            y1(1)       =  y1(2)
390         continue
400         jb          =  jl + 1
c
c       If the number of intersection points,  icnt , is less than or
c       equal to one, skip the sort procedure.  Otherwise, sort the
c       intersection points in ascending order for use in plotting the
c       shaded rays.  Then delete those duplicate values of  ysav .
```

```
c
          if(icnt .le. 1)  go to 500
          call dsort(ysav,dir,icnt)
          idx       =  1
          do 450 i=2,icnt
          if(abs(ysav(i)-ysav(i-1)) .gt. tol)  go to 440
c
c     A duplicate intersection point has been encountered.
c     Discard the duplicate point, but update the  dir  array
c     so that if one of the segments was vertical, the value
c     of  dir  will indicate to which side of the vertical ray
c     the nonvertical segment lies.
c
c
          dir(idx)  =  dir(i-1) + dir(i)
c
c     If the following test is satisfied, both line segments
c     producing the duplicate intersection point lie on the
c     same side of the vertical ray.  Do not count them when
c     determining the parity.
c
          if(abs(dir(idx)) .gt. 1.5)  ispar(idx) = 0
c
c     If one of the duplicate intersection points resulted from
c     a vertical line segment, set the value of  ispar  to  -1 .
c     The absolute value of  ispar  will be used later to deter-
c     mine the parity for shading the vertical ray.
c
          if(abs(abs(dir(idx)) - 1.0) .le. tol)  ispar(idx) = -1
          if(idx .le. 1)  go to 450
c
c     If the previous duplicate value that was loaded came from
c     a vertex with a vertical line segment and this duplicate
c     intersection value also has a vertical line segment, then
c     test to see if the nonvertical line segments creating the
c     vertices on the vertical ray intersect the ray from opposite
c     sides.  If so, set the value of  ispar  to 2 so that the
c     parity does not change at the second intersection.  If
c     the two line segments enter from the same side, keep the
c     parity odd.
c
          itest     =  ispar(idx-1) + ispar(idx)
          test      =  dir(idx-1)*dir(idx)
          if(itest .eq. -2  .and.  test .gt. 0)  ispar(idx) = 1
          if(itest .eq. -2  .and.  test .lt. 0)  ispar(idx) = 2
          go to 450
440       idx       =  idx + 1
          ysav(idx) =  ysav(i)
          dir(idx)  =  dir(i)
          ispar(idx)=  1
450       continue
          icnt      =  idx
c
c     The line segments in the vertical ray are tested for parity
c     and plotted accordingly.
c
500       itest     =  ipar(is)
          if(icnt .le. 0)  go to 560
          ll        =  0
          do 550 l=1,icnt
```

```
      if(ispar(l) .eq. 0)  go to 550
      ys          =  ysav(l)
      ll          =  ll + iabs(ispar(l))
      itest       =  mod(ll+ipar(is),2)
      if(ys .lt. ymin)  go to 550
      if(ys .gt. ymax)  ys = ymax
      if(itest .eq. 0)  call drawa(xx,ys)
      if(itest .eq. 1)  call movea(xx,ys)
550   continue
560   if(itest .eq. 1)  call drawa(xx,ymax)
600   continue
      call anmode
      return
      end
```

```
      subroutine shift(is)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /bound/     xb(20000,2),   yb(20000,2),   nch(2000,2),
                         noch(2),       kod(2000,2),   ipar(2),
                         icnd(2),       kap(15,2),     nptot(2)
c
c     This subroutine is designed to shift the x,y-boundary coordinates
c     from the front of the xb,yb-arrays to the end of the xb,yb-arrays.
c     The old chains are processed in order and the new chains are
c     overwritten onto the front of the arrays.  This is done to make
c     the most efficient use of the reserved array space.
c
c     If the storage of these new chains requires the overwriting of
c     that portion of the array where the old chains are stored, no
c     harm will be done because only those old chains which have already
c     been processed will be overwritten.
c
c     A definition of the common block variables is provided in
c     subroutine  chain .
c
          iend    =  nptot(is)
          ibeg    =  20000 - iend
          do 10 i=1,iend
          xb(ibeg+i,is)=xb(i,is)
          yb(ibeg+i,is)=yb(i,is)
10        continue
          return
          end
```

```
      subroutine sing(iwhy)
c
c   PRINTS ERROR MESSAGE FOR SINGULAR MATRIX
c
11    format("  MATRIX WITH ZERO ROW IN DECOMPOSE")
12    format("  SINGULAR MATRIX IN DECOMPOSE.  ZERO DIVIDE IN SOLVE")
      go to (1,2),iwhy
1     print 11
      return
2     print 12
      return
      end
```

```
      subroutine slice(xmin,xmax,is)
c
c           This subroutine was written and programmed by
c                          A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /bound/ xb(20000,2),   yb(20000,2),   nch(2000,2),
                     noch(2),       kod(2000,2),   ipar(2),
                     icnd(2),       kap(15,2),     nptot(2)
      common /chain/ icf(100),      icl(100),      xint(101),
                     iptf(2000),    iptl(2000)
c
c     Initialize the first (icf) and the last (icl) chain index
c     to zero for each of the twenty slices.
c
      do 10 i=1,100
      icf(i)    =  0
10    icl(i)    =  0
c
c     Initialize the indices of the first point in a sliced chain
c     (iptf) and the last point in a sliced chain (iptl) to zero.
c
      do 20 i=1,2000
      iptf(i)   =  0
20    iptl(i)   =  0
c
c     Initialize the sliced chain count (lch) and the sliced chain
c     saved count (lsv) to zero, load the number of chains into
c     nis , and establish the x-limits (xint) of the vertical slices.
c
      lch       =  0
      lsv       =  0
      nis       =  noch(is)
      xdel      =  .01*(xmax - xmin)
      xint(1)   =  xmin
      xint(101) =  xmax
      do 30 i=2,100
      qi        =  i - 1
30    xint(i)   =  xmin + qi*xdel
c
c     Step through each of the twenty vertical slices and index
c     those portions of the boundary chains having line segments
c     partially or totally within the slice.
c
      do 800 l=1,100
      xm1       =  xint(l)
      xm2       =  xint(l+1)
      jl        =  0
```

```
c
c     Scan through each of the  nis  chains contained in the is-th
c     boundary array to determine if any part of that chain enters
c     the (xm1,xm2)-slice interval of the l-th slice.
c
          do 700 i=1,nis
          ibeg      =  0
          jb        =  jl + 1
          jl        =  jl + iabs(ncn(i,is))
          xx        =  xb(jb,is)
          test      =  (xm1 - xx)*(xx - xm2)
          if(test .lt. 0.)  go to 100
c
c     If the value of  test  is greater than or equal to zero, the
c     first point of a boundary chain lies within the slice interval.
c     Set  ibeg  to one, increment the sliced chain counter (lch) and
c     store the index of the point in the  iptf -array.
c
          ibeg      =  1
          lch       =  lch + 1
          iptf(lch) =  jb
c
c     Continue to test the remaining points in the  nis-th chain to
c     determine if they enter or exit the  l-th slice.
c
100       jbp       =  jb + 1
          do 600 j=jbp,jl
c
c     If the  j-th boundary segment completely spans the (xm1,xm2)-slice
c     interval, proceed to the logic for establishing the indices for
c     a two point chain.
c
          xxj       =  xb(j,is)
          xxm       =  xb(j-1,is)
          test1     =  (xxm - xm1)*(xm1 - xxj)
          test2     =  (xxm - xm2)*(xm2 - xxj)
          if(test1 .gt. 0.  .and.  test2 .gt. 0.)  go to 200
c
c     If the boundary vertex lies outside of the (xm1,xm2)-slice interval,
c     proceed to the logic to check the segment to see if it exits the
c     slice.
c
          test      =  (xm1 - xxj)*(xxj - xm2)
          if(test .lt. 0.)  go to 400
c
c     If this is not the first point of the subchain to enter the
c     (xm1,xm2)-slice interval, and if it does not exit from the
c     interval, then skip to the logic to determine if it is the last
c     point in the chain.
c
          if(ibeg .eq. 1)  go to 300
c
c     A boundary chain segment has entered the vertical slice, or else
c     totally contains the vertical slice.  Reset  ibeg , increment
c     the chain count (lch), and store the index of the first endpoint
c     of the line segment into the  iptf -array, designating it as the
c     first point in the  lch-th sliced chain.
c
200       ibeg      =  1
```

```
            lch       =  lcn + 1
            iptf(lcn) =  j - 1
            if(test1 .gt. 0.  .and.  test2 .gt. 0.)  go to 500
c
c       The second endpoint of this boundary segment lies within the
c       slice interval.  Test to see if it is the last point in the chain,
c       in which case, set the indices and reset  ibeg  to zero.  Other-
c       wise, continue testing the points in the chain.
c
300         if(j .eq. jl)  go to 500
            go to 600
c
c       If the line segment is totally outside of the slice interval, skip
c       to the end of the loop and test the next point in the boundary
c       chain.  Otherwise, set the  iptl  index to reference the point
c       exiting the slice interval.
c
400         if(ibeg .eq. 0)  go to 600
500         ibeg      =  0
c
c       The sign of  iptl  is set to the sign of  nch  of the chain
c       from which that subchain was derived.  This is to flag those
c       appended chains derived from the  complete  routine.  This is
c       so they will not be used for determining boundary intersection
c       points.
c
            iptl(lcn) =  isign(j,nch(i,is))
600         continue
700         continue
c
c       Store the indices for the first (icf) and for the last (icl)
c       chain in the  l-th slice interval, reset  lsv  and continue.
c
            icf(l)    =  lsv + 1
            icl(l)    =  lch
            lsv       =  lcn
800         continue
            return
            end
```

```
      subroutine smooth(xx,yy,ic)
c
c           This subroutine was written and programmed by
c                          A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
c
c   THIS SUBROUTINE KEEPS TRACK OF THREE SEQUENTIAL POINTS BELONGING
c   TO THE CURVE FOR THE PURPOSE OF SMOOTHING BY INTERPOLATION.
c   AFTER THE INTERPOLATION STEP IS COMPLETED, THE THREE POINTS ARE
c   SHIFTED WITH THE MIDDLE POINT OF THE PREVIOUS STEP BECOMING THE
c   FIRST POINT, THE LAST POINT BECOMING THE MIDDLE POINT, AND THE
c   NEW POINT BECOMING THE LAST POINT.
c   (XF,YF)  =  FIRST POINT
c   (XM,YM)  =  MIDDLE POINT
c   (XL,YL)  =  LAST POINT
c   (XX,YY)  =  NEW POINT
c   THE PROCEDURE ADJUSTS THE SLOPES OF THE FIRST AND THE LAST POINTS
c   OF NON-CLOSED CURVES.  IT ALSO SAVES THE NECESSARY INFORMATION
c   FOR SMOOTHING AT THE FIRST AND LAST POINTS OF THE CLOSED CURVES.
c   THE DIFFERENCES BETWEEN SEQUENTIAL VALUES OF  X  AND  Y , ARE
c   STORED IN VARIABLES WITH  U  AND  V  DESIGNATIONS.
c   NOTE:  IC  =  -1   INDICATES LAST POINT OF A CLOSED CURVE
c          IC  =   0   INDICATES LAST POINT OF A NON-CLOSED CURVE
c          IC  =   1   INDICATES FIRST POINT OF CURVE.
c
          if(ic .ge. 3) go to 20
          if(ic .le. 0) go to 50
          if(ic .ne. 1) go to 10
c
c   INITIALIZE FOR FIRST POINT OF CURVE
c
          xf    =  xx
          yf    =  yy
          return
c
c   INITIALIZE FOR SECOND POINT OF CURVE
c
10        xm    =  xx
          ym    =  yy
          return
c
c   SECCESSIVE POINTS ON THE CURVE ARE INSERTED INTO THE LAST
c   INTERPOLATION POINT, (XL,YL).  IF (XL,YL) IS IDENTICAL TO
c   (XM,YM) RETURN WITHOUT PROCEEDING WITH INTERPOLATION.
c
20        xl    =  xx
          yl    =  yy
```

```
c
c    WITH INFORMATION AVAILABLE FOR FIRST THREE POINTS ON CURVE,
c    COMPUTE AND ADJUST POINT DIFFERENCES  U1  AND  V1  BEFORE
c    PROCEEDING WITH INTERPOLATION.
c
30          if(ic .ge. 4) go to 40
            int    =  1
            u      =  xm - xf
            v      =  ym - yf
            u1     =  xl - xm
            v1     =  yl - ym
            call adjust(u,v,u1,v1)
c
c    SAVE THE POINT DIFFERENCE VALUES FOR FURTHER USE IF CURVE IS
c    CLOSED.
c
            su     =  u
            sv     =  v
c
c    PERFORM CUBIC SPLINE INTERPOLATION BETWEEN (XF,YF) AND (XM,YM)
c
40          u2     =  xl - xm
            v2     =  yl - ym
            if(abs(u2) + abs(v2) .ge. .0000005)  go to 45
            u2       =  u1
            v2       =  v1
            go to 46
45          call spline(xf,yf,xm,ym,u1,v1,u2,v2,int)
c
c    SHIFT POINT VALUES BEFORE ADDING NEW POINT FOR NEXT INTERPOLATION
c    STEP.
c
46          u1     =  u
            v1     =  v
            u      =  u2
            v      =  v2
            xf     =  xm
            yf     =  ym
            xm     =  xl
            ym     =  yl
            return
c
c    MAKE ADJUSTMENTS FOR INTERPOLATION TO LAST POINT OF CURVE.
c
50          i      =  1
            xl     =  xx
            yl     =  yy
            u2     =  xl - xm
            v2     =  yl - ym
c
c    IF CURVE IS CLOSED, TRANSFER TO CLOSED CURVE LOGIC.  OTHERWISE
c    MAKE ADJUSTMENTS FOR CURVATURE AT LAST INTERVAL.
c
            if(ic .lt. 0) go to 60
            su     =  xm - xf
            sv     =  ym - yf
            call adjust(u2,v2,su,sv)
c
c    MAKE INTERPOLATION FOR NEXT TO LAST INTERVAL.  THEN SHIFT
c    VALUES FOR INTERPOLATION THROUGH LAST INTERVAL.
```

```
c
60          if(i .eq. 1) go to 70
            u2     =  su
            v2     =  sv
70          call spline(xf,yf,xm,ym,u1,v1,u2,v2,int)
c
c     MAKE ONE ADDITIONAL SHIFT AND REPEAT THE INTERPOLATION PROCEDURE,
c     THIS TIME FOR THE LAST INTERVAL OF THE CURVE.
c
            if(i .ge. 2) return
            xf     =  xm
            yf     =  ym
            xm     =  xl
            ym     =  yl
            u1     =  u
            v1     =  v
            i      =  i + 1
            go to 60
            end
```

```
      subroutine solve(ul,b,x,nm,na)
c
c   BACK SOLVES FOR THE SOLUTION VECTOR AFTER DECOMPOSITION
c
      dimension      ul(na,1),      b(1),          x(1),
                     ips(100)
      common /ir/ ips
      n        =  nm
      np1      =  n + 1
      ip       =  ips(1)
      x(1)     =  b(ip)
      do 2 i=2,n
      ip       =  ips(i)
      im1      =  i - 1
      sum      =  0.
      do 1 j=1,im1
1     sum      =  sum + ul(ip,j)*x(j)
2     x(i)     =  b(ip) - sum
      ip       =  ips(n)
      x(n)     =  x(n)/ul(ip,n)
      do 4 iback=2,n
      i        =  np1 - iback
      ip       =  ips(i)
      ip1      =  i + 1
      sum      =  0.
      do 3 j=ip1,n
3     sum      =  sum + ul(ip,j)*x(j)
4     x(i)     =  (x(i) - sum)/ul(ip,i)
      return
      end
```

```
      subroutine sort(xq,n)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      dimension xq(n)
      nm          =  n - 1
      do 200 i=1,nm
      xs          =  xq(i)
      js          =  i
      ip          =  i + 1
      do 100 j=ip,n
      if(xq(j) .ge. xs)  go to 100
      xs          =  xq(j)
      js          =  j
100   continue
      if(js .le. i)  go to 200
      xq(js)      =  xq(i)
      xq(i)       =  xs
200   continue
      return
      end
```

```
      subroutine spline(xf,yf,xm,ym,u1,v1,u2,v2,int)
c
c           This subroutine was written and programmed by
c                              A. C. Olson
c           for use in the GARNET interactive graphics system.
c           GARNET was developed to perform resource mapping
c           and resource estimations for the NATIONAL COAL
c           RESOURCES DATA SYSTEM.
c
c
      common /graphc/     xmin,          ymin,          zmin,
     1     zgrid(65000),  xmax,          ymax,          zmax,
     2     istat(65000),  nx,            ny,            nz,
     3     x(16000),      nxgd,          nygd,          ncell,
     4     y(16000),      delgrd,        igrid,         icell,
     5     into,          zint,          zlevel,        level,
     6     nbold,         lplot,         dashl,         pcrang,
     7     levmin,        levmax,        ifit,          isw1,
     8     isw2,          isw3,          ismth,         icnt
c
c    THIS ALGORITHM COMPUTES THE CUBIC COEFFICIENTS FOR A PSUEDO-
c    SPLINE CURVE AND USES THIS CURVE TO INTERPOLATE XY-COORDINATE
c    VALUES BETWEEN THE ENDPOINTS OF EACH INTERVAL.
c    UNLIKE NORMAL SPLINE PROCEDURES, THE TECHNIQUE USED HERE DOES
c    NOT REQUIRE THE SECOND DERIVATIVES OF TWO ADJACENT SPLINE
c    CURVES TO BE EQUAL AT THE SPLINE JUNCTION.
c    THE PARAMETRIC CUBIC EQUATIONS USED IN THIS SPLINE SYSTEM
c    ARE OF THE FORM:
c                        X  =  AX*T**3 + BX*T**2 + CX*T + DX , AND
c                        Y  =  AY*T**3 + BY*T**2 + CY*T + DY .
c    WHERE AX, BX, CX, AND DX DENOTE THE COEFFICIENTS FOR THE
c    EQUATION FOR  X  AS A FUNCTION OF THE PARAMETER  T .  AND
c    AY, BY, CY, AND DY REPRESENT THE COEFFICIENTS FOR THE
c    EQUATION FOR  Y  AS A FUNCTION OF THE SAME PARAMETER  T .
c    GIVEN THREE POINTS, FIRST (XF,YF), MIDDLE (XM,YM), AND
c    LAST (XL,YL) AND THREE DIFFERENCE VECTORS, (U1,V1) BETWEEN
c    (XF,YF) AND THE POINT WHICH WAS (XF,YF) IN THE PRECEDING
c    SPLINE INTERVAL, (UM,VM) BETWEEN (XM,YM) AND (XF,YF). AMD
c    (U2,V2) BETWEEN (XL,YL) AND (XM,YM), THIS PROCEDURE
c    ESTIMATES THE DERIVATIVES AT THE END POINTS OF THE SPLINE
c    INTERVAL AND COMPUTES THE COEFFICIENTS FOR EACH OF THE
c    PARAMETRIC EQUATIONS.  THE PARAMETER  T  IS DEFINED TO BE
c    EQUAL TO ZERO AT (XF,YF) AND EQUAL TO ONE AT (XM,YM).
c    THE FOLLOWING VALUES MUST BE COMPUTED FOR THE FIRST SPLINE
c    INTERVAL.  FOR FOLLOWING INTERVALS, THE VALUES ARE SIMPLY
c    SHIFTED FROM RIGHT TO LEFT AND THEN NEW DIFFERENCE VECTOR
c    INFORMATION IS COMPUTED FOR THE RIGHT SIDE FOR EACH
c    SUCCEEDING INTERVAL.
c
      if(int .gt. 1) go to 50
      icnt  =  1
      int   =  2
```

```
          um     =  xm - xf
          vm     =  ym - yf
c
c    THE FOLLOWING STATEMENTS COMPUTE THE VALUES OF THE VECTORS
c    WHICH ARE ESTIMATES OF THE SLOPES AT THE POINTS (XF,YF) AND
c    (XM,YM).
c    IF  D1  IS THE LENGTH OF (U1,V1),  D2  THE LENGTH OF (UM,VM),
c    AND  D3  THE LENGTH OF (U2,V2), THEN
c      U1P  =  (UM*D1/D2 + U1*D2/D1), AND
c      V1P  =  (VM*D1/D2 + V1*D2/D1).
c    THE VECTOR (U1P,V1P) IS THEN NORMALIZED TO UNITY.  THE
c    STATEMENTS COMPUTE THE NORMALIZED (U1P,V1P) VECTOR.
c    THE CODE IS WRITTEN IN A SLIGHTLY DIFFERENT FORM, SO THAT
c    A COMPARISON IS NOT READILY OBVIOUS.
c    D1S  IS THE SQUARE OF THE LENGTH OF (U1,V1),
c    D2S  IS THE SQUARE OF THE LENGTH OF (UM,VM), AND
c    D3S  IS THE SQUARE OF THE LENGTH OF (U2,V2).
c
          d1s    =  u1**2 + v1**2
          d2s    =  um**2 + vm**2
          u1p    =  d1s*um + d2s*u1
          v1p    =  d1s*vm + d2s*v1
          dum    =  sqrt(u1p**2 + v1p**2)
          div    =  1.0
          if (dum .gt. .0) div = 1.0/dum
          u1p    =  div*u1p
          v1p    =  div*v1p
c
c    DP1  IS THE MAGNITUDE OF THE DOT PRODUCT BETWEEN THE VECTORS
c    (UM,VM) AND (U1P,V1P).
c    SINCE THE MAGNITUDE OF (U1P,V1P) IS UNITY,  DP1  IS SIMPLY
c    THE LENGTH OF THE COMPONENT OF (UM,VM) IN THE (U1P,V1P)
c    DIRECTION.
c    THE (U1P,V1P) VECTOR IS THEN SCALED BY  DP1 , TO PRODUCE
c    (UU1P,VV1P).  THE COMPONENTS OF THIS VECTOR REPRESENT THE
c    ESTIMATED DERIVATIVE OF  X  WITH RESPECT TO  T  (I. E.
c    DX/DT  =  UU1P) AND THE DERIVATIVE OF  Y  (DY/DT  =  VV1P)
c    WITH RESPECT TO  T  AT THE POINT (XF,YF).
c    FOR THE FIRST INTERVAL OF THE SPLINE CURVE, THE VALUE OF  INT
c    IS SET TO  1  IN THE CALLING PROGRAM  "SMOOTH."  THE PRECEDING
c    STATEMENTS ARE NECESSARY ONLY TO OBTAIN THE VECTOR VALUES FOR
c    THE LEFT SIDE FOR THE FIRST SPLINE INTERVAL.  AFTER THE FIRST
c    INTERVAL, THE VECTOR VALUES FOR THE RIGHT SIDE ARE SHIFTED
c    INTO THE VARIABLES FOR THE LEFT SIDE, AND SINCE  INT  IS NOW
c    SET TO  2 , THE PRECEDING STATEMENTS WILL BE SKIPPED UNTIL A
c    NEW CURVE IS STARTED.
c
50        dp1    =  abs(u1p*um + v1p*vm)
          if(dp1 .le. 0.0) dp1 = 1.0
          uu1p   =  dp1*u1p
          vv1p   =  dp1*v1p
c
c    THE FOLLOWING STATEMENTS COMPUTE THE VALUE OF (UU2P,VV2P) IN
c    SAME MANNER AS (UU1P,VV1P) WAS COMPUTED ABOVE.
c
          d3s    =  u2**2 + v2**2
          u2p    =  d2s*u2 + d3s*um
          v2p    =  d2s*v2 + d3s*vm
          dum    =  sqrt(u2p**2 + v2p**2)
          div    =  1.0
```

```
            if (dum .gt. .0) div = 1.0/dum
            u2p    =  div*u2p
            v2p    =  div*v2p
            dp2    =  abs(u2p*um + v2p*vm)
            if(dp2 .le. 0.0) dp2 = 1.0
            uu2p   =  dp2*u2p
            vv2p   =  dp2*v2p
c
c     THE SLOPE VECTOR VALUES ARE USED TO CALCULATE THE COEFFICIENTS
c     AX, BX, AY, AND BY.  BY THE OTHER SPLINE BOUNDARY CONDITIONS,
c           CX   =  UU1P
c           CY   =  VV1P
c           DX   =  XF
c           DY   =  YF
c     AND THESE VALUES ARE SUBSTITUTED DIRECTLY INTO THE CUBIC
c     EQUATIONS BELOW.
c
            ax     =  uu2p + uu1p - 2.0*um
            bx     =  um   - uu1p - ax
            ay     =  vv2p + vv1p - 2.0*vm
            by     =  vm   - vv1p - ay
c
c     THE LENGTH  DP  IS THE TOTAL PARAMETRIC LENGTH BETWEEN (XF,YF)
c     AND (XM,YM) AS THE PARAMETER  T  GOES FROM  0  TO  1 .
c     THE FOLLOWING STATEMENTS DIVIDE THE PARAMETRIC LENGTH INTO
c     8*DP SUBINTERVALS OF EQUAL LENGTH.
c
            dp     =  dp1 + dp2
            a      =  8.0*dp + 1.0
            n      =  a
            if(n .gt. 6)  n = 6
            if(n .le. 1) go to 300
            a      =  n
            d      =  1.0/a
c
c     THE FOLLOWING LOOP COMPUTES THE XY-COORDINATES FOR EACH OF
c     THE INTERPOLATED POINTS WITHIN THE FITTED SPLINE INTERVAL.
c
            do 100 i=1,n
            qi     =  i
            t      =  qi*a
            if(t .gt. 1.0) go to 100
            icnt   =  icnt + 1
            x(icnt)  =  ((ax*t + bx)*t + uu1p)*t + xf
            y(icnt)  =  ((ay*t + by)*t + vv1p)*t + yf
100         continue
c
c     AFTER ALL OF THE COORDINATES HAVE BEEN COMPUTED FOR THAT
c     SPLINE INTERVAL, THE VECTOR VALUES FOR THE RIGHT SIDE ARE
c     SHIFTED INTO THE VARIABLES FOR THE LEFT SIDE AND THEN CONTROL
c     IS RETURNED TO  "SMOOTH"  WHICH WILL AGAIN CALL  "SPLINE"  IF
c     THERE ARE ADDITIONAL INTERVALS LEFT TO BE SMOOTHED.
c
200         u1p    =  u2p
            v1p    =  v2p
            d1s    =  d2s
            d2s    =  d3s
            um     =  u2
            vm     =  v2
            return
```

```
300       icnt  =  icnt + 1
          x(icnt)  =  xm
          y(icnt)  =  ym
          go to 200
          end
```

```
      subroutine surf(nctot,fls,dfls,ddfls,c)
c
c          This subroutine was written and programmed by
c                              A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      double precision    dfls(69),      ddfls(69,69),
     1    fls,           df(69),        c(69)
      double precision    zp(1500)
      common /surfac/     xob(1500),     yob(1500),
     1    zt(1500),      zob(1500),     residu(1500),
     2    m,             n,             np,
     3    perd,          resum,         ntc,
     4    npc,           msq,           msq2,
     5    msq3,          msq4,          sig
      fls    =  0.
      resum  =  0.
      do 50 i=1,nctot
      dfls(i)  =  0.
      do 50 j=1,nctot
50    ddfls(i,j)  =  0.
      df(nctot)  =  1.
      do 500 l=1,np
      zp(l)  =  c(nctot)
      if(ntc .le. 0) go to 250
      do 200 i=1,m
      qi     =  i
      im     =  (i-1)*m
      qipd   =  qi*perd
      sx     =  sin(qipd*xob(l))
      cx     =  cos(qipd*xob(l))
      syi    =  sin(qipd*yob(l))
      cyi    =  cos(qipd*yob(l))
      do 100 j=1,m
      qj     =  j
      qjpd   =  qj*perd
      sy     =  sin(qjpd*yob(l))
      cy     =  cos(qjpd*yob(l))
      im1    =  im  + j
      im2    =  im1 + msq
      im3    =  im1 + msq2
      im4    =  im1 + msq3
      zp(l)  =  zp(l) + c(im1)*sx*sy + c(im2)*sx*cy
     1                + c(im3)*cx*sy + c(im4)*cx*cy
      df(im1)  =  sx*sy
      df(im2)  =  sx*cy
      df(im3)  =  cx*sy
      df(im4)  =  cx*cy
```

```
100       continue
          mm1    =  msq4 + i
          mm2    =  mm1 + m
          mm3    =  mm2 + m
          mm4    =  mm3 + m
          zp(l)  =  zp(l) + c(mm1)*sx  + c(mm2)*cx
                          + c(mm3)*syi + c(mm4)*cyi
          df(mm1)  =  sx
          df(mm2)  =  cx
          df(mm3)  =  syi
          df(mm4)  =  cyi
200       continue
250       continue
          if(npc .le. 0) go to 450
          nsum  =  ntc
          do 400 j=1,n
          nn  =  n + 2 - j
          do 300 i=1,nn
          tx       =  xob(l)
          ty       =  yob(l)
          ix       =  i - 1
          jy       =  nn - i
          if(ix .eq. 0)  tx = 1.
          if(jy .eq. 0)  ty = 1.
          term     =  tx**ix*ty**jy
          zp(l)  =  zp(l) + c(nsum+i)*term
          df(nsum+i)  =  term
300       continue
          nsum  =  nsum + nn
400       continue
          nsum  =  nsum - nn
450       continue
          residu(l)  =  zob(l) - zp(l)
          resum  =  resum + residu(l)
          fls  =  fls + residu(l)**2
          do 475 i=1,nctot
          dfls(i)  =  dfls(i) - 2.*residu(l)*df(i)
          do 475 j=1,nctot
475       ddfls(i,j)  =  ddfls(i,j) + 2.*df(i)*df(j)
500       continue
          do 600 l=1,np
600       zt(l)  =  zp(l)
          return
          end
```

```
      subroutine surfit(c)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      double precision          dfls(69),      ddfls(69,69),
                  fls,          cnorm,         dnorm,
                  cc(69),       cnsav,         fsav
      dimension   c(69)
      common /hold/ ddfls
      common /surfac/     xob(1500),     yob(1500),
           zt(1500),      zob(1500),     residu(1500),
           m,             n,             np,
           perd,          resum,         ntc,
           npc,           msq,           msq2,
           msq3,          msq4,          sig
      irite    =  2
      pi       =  3.1415926535
      m      =  2
      n      =  2
30    call promot("DO YOU WANT TO USE NON-STANDARD SURFACE PARAMETERS? yes OR no: ",63
      read (5,1001)itest
1001  format(a4)
      if(itest .ne. "yes" .and. itest .ne. "no") go to 30
      if(itest .eq. "no") go to 40
      call prompt("ENTER DEGREE OF POLYNOMIAL: ",28)
      read(5,1005,end=9999)n
1005  format(v)
      call prompt("ENTER ORDER OF TRIGONOMETRIC SERIES: ",37)
      read(5,1005,end=9999)m
      scale    =  1.
      if(m .le. 0) go to 60
40    lmax   =  1
      lmin   =  1
      do 50 i=2,np
      if(zob(i) .gt. zob(lmax)) lmax = i
50    if(zob(i) .lt. zob(lmin)) lmin = i
      perd   =  sqrt((xob(lmax) - xob(lmin))**2
                  + (yob(lmax) - yob(lmin))**2)
      period  =  2.*perd
      perd    =  pi/perd
      scale   =  2./(zob(lmin) + zob(lmax))
60    if(irite .gt. 0) write(6,1002) period,scale
1002  format(/" THE FUNDAMENTAL FREQUENCY OF THE TRIGONOMETRIC"
      " EXPANSION IS ",f7.3," "/" (GIVEN IN THE X-Y UNITS)"//
      " THE COMPUTED SCALE FACTOR IS ",e9.3," "/)
      ntc    =  4*m*(m+1)
```

```
      npc    =  (n+1)*(n+2)/2 - 1
      nctot  =  ntc + npc + 1
      msq    =  m**2
      msq2   =  msq  + msq
      msq3   =  msq2 + msq
      msq4   =  msq3 + msq
      nsum   =  ntc
      fls    =  1.e8
      cnsav  =  1.d25
      do 100 i=1,ntc
100   cc(i)  =  10.d0
      do 120 j=1,n
      nn     =  n + 2 - j
      div    =  20.**(1-j)
      do 110 i=1,nn
110   cc(nsum+i)  =  div
      div    =  20.*div
      nsum   =  nsum + nn
120   continue
      cc(nctot)   =  zob(1)
      do 200 i=1,np
200   zob(i)  =  zob(i)*scale
5000  continue
      fsav   =  fls
      call surf(nctot,fls,dfls,ddfls,cc)
      if(abs(1.d0 - fls/fsav) .lt. .0001)  go to 5300
      call dmatin(nctot,ddfls)
      cnorm  =  0.
      dnorm  =  0.
      do 5200 i=1,nctot
      do 5100 j=1,nctot
5100  cc(i)  =  cc(i) - dfls(j)*ddfls(j,i)
      cnorm  =  cnorm + cc(i)**2
5200  dnorm  =  dnorm + dfls(i)**2
      cnorm  =  sqrt(cnorm)
      dnorm  =  sqrt(dnorm)
      if(irite .gt. 1) write(6,1003) fls,cnorm,dnorm
1003  format(1x,"FLS = ",e12.7,5x,"CNORM = ",e12.7,5x,"DNORM = "
     ,e12.7)
      if(abs(1.d0 - cnorm/cnsav) .lt. .0001)  go to 5300
      cnsav  =  cnorm
      if(dnorm .gt. .0005) go to 5000
5300  xxx    =  np - nctot - 1
      sig    =  sqrt(fls/xxx)/scale
      do 300 l=1,np
      zob(l)  =  zob(l)/scale
300   residu(l)  =  residu(l)/scale
      do 350 i=1,nctot
350   c(i)  =  cc(i)/scale
      if(irite .gt. 2)  write(6,1004) (c(i),i=1,nctot)
      if(irite .gt. 0)  write(6,1007)
1007  format(/" THE TREND SURFACE HAS CONVERGED TO A SOLUTION ..."/
           " PRESS THE CARRIAGE RETURN WHEN READY TO PROCEED.")
      if(irite .gt. 0)  read(5,1006) idumm
1006  format(a1)
1004  format(5(1x,e13.5))
9999  return
      end
```

```
      subroutine tsort(aq,bq,cq,n)
c
c          This subroutine was written and programmed by
c                              A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      dimension       aq(n),          bq(n),          cq(n)
      nm          =   n - 1
      do 200 i=1,nm
      as          =   aq(i)
      js          =   i
      ip          =   i + 1
      do 100 j=ip,n
      if(aq(j) .ge. as)  go to 100
      as          =   aq(j)
      js          =   j
100   continue
      if(js .le. i)  go to 200
      aq(js)      =   aq(i)
      aq(i)       =   as
      bs          =   bq(js)
      bq(js)      =   bq(i)
      bq(i)       =   bs
      cs          =   cq(js)
      cq(js)      =   cq(i)
      cq(i)       =   cs
200   continue
      return
      end
```

```
      subroutine volume(radrel,vol)
c
c          This subroutine was written and programmed by
c                          A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /surfac/     xob(1500),     yob(1500),
     1    zt(1500),      zob(1500),     residu(1500),
     2    m,             n,             np,
     3    perd,          resum,         ntc,
     4    npc,           msq,           msq2,
     5    msq3,          msq4,          sig
      common /graphc/     xmin,          ymin,          zmin,
     1    zgrid(65000),  xmax,          ymax,          zmax,
     2    istat(65000),  nx,            ny,            nz,
     3    x(16000),      nxgd,          nygd,          ncell,
     4    y(16000),      delgrd,        igrid,         icell,
     5    into,          zint,          zlevel,        level,
     6    nbold,         lplot,         dashl,         pcrang,
     7    levmin,        levmax,        ifit,          isw1,
     8    isw2,          isw3,          isw4,          isw5
      common /ondpt/      yp(4,50),      nc(4),         ibnd,
     1                    ypmin,         ypmax
      dimension           xc(4),         yr(4),         inout(4)
      double precision    yzarea(2001),  vol,           zl,
     1    dx,            dyl,           dyu,           zu,
     2    a,             b,             c,             d,
     3    z1,            z2,            z3,            z4
      call delim(nx1,nx2,ny1,ny2)
      nym1     =  ny2 - 1
      iray     =  4*(nx2 - nx1 + 1) - 3
      do  10 i=1,iray
10    yzarea(i)  =  0.
      d4       =  .25*delgrd
      do 150 i=nx1,nx2
      qi       =  i - 2
      xx       =  qi*delgrd + xmin
      xc(1)    =  xx + d4
      xc(2)    =  xc(1) + d4
      xc(3)    =  xc(2) + d4
      xc(4)    =  xc(3) + d4
      call cirobs(xc(1),xc(4),radrel,$150)
      ypmax    =  ymin
      ypmin    =  ymax
      call bndpnt(xc)
      ncsum    =  0
      do 20 l=1,4
      ncsum    =  ncsum + nc(l)
```

```
20        inout(l)  =  0
          if(ncsum .eq. 0)  go to 150
          ny1p      =  (ypmin - ymin + delgrd*1.00001)/delgrd
          ny2p      =  (ypmax - ymin + delgrd*1.00001)/delgrd
          do 100 j=ny1p,ny2p
          qj        =  j - 1
          yy        =  qj * delgrd + ymin
          yr(1)     =  yy + d4
          yr(2)     =  yr(1) + d4
          yr(3)     =  yr(2) + d4
          yr(4)     =  yr(3) + d4
          call cirobs$ciry(yy,yr(4),radrel,$100)
          kgrid     =  (i - 2)*ny + j
          kgdny     =  kgrid + ny
          if(kgrid .le. 0) kgrid = j
          z1        =  zgrid(kgrid)
          z2        =  zgrid(kgrid+1)
          z3        =  zgrid(kgdny+1)
          z4        =  zgrid(kgdny)
          if(abs(z1) + abs(z2) + abs(z3) + abs(z4)  .le. 0.) go to 100
          a         =  z1
          b         =  (z4 - z1)/delgrd
          c         =  (z2 - z1)/delgrd
          d         =  (z1 - z2 + z3 - z4)/delgrd**2
          do  80 ii=1,4
          if(nc(ii) .eq. 0)  go to 80
          if(i .eq. nx1  .and.  ii .ne. 4) go to 80
          iray      =  (i - nx1 - 1)*4 + ii + 1
          yyp       =  yy
          xt        =  xc(ii)
      call movea(xt,yy)
          do 70 jj=1,4
          yl        =  yyp
          yu        =  yr(jj)
          call cirobs$cirtst(xt,yu,dmin,xxmin,yymin,radrel,$70)
          call bndpnt$inpnt(yl,yu,yq,ii,inreg,iflag)
          if(inreg .eq. 0  .and.  inout(ii) .eq. 0)  go to 70
          yt        =  yr(jj)
          if(inreg .eq. 0)  yt = yyp
          if(dmin .gt. radrel  .and.  inout(ii) .eq. 0)  go to 70
          if(dmin .le. radrel  .and.  inout(ii) .eq. 1  .and.
             inreg .eq. 1)  go to 50
          call cirint$vlcir(xt,yl,xt,yu,xxmin,yymin,
                            radrel,x1,y1,x2,y2,int)
          if(y1 .lt. yl)  y1 = yl
          if(y2 .gt. yu)  y2 = yu
          if(inout(ii) .eq. 1)  go to 40
          if(iflag .eq. 1  .and.  (yl-yq)*(yq-yu) .ge. 0.)  yl = yq
          if( int  .eq. 1  .and.  (yl-y1)*(y1-yu) .ge. 0.)  yl = y1
          inout(ii)  =  1
      call dasha(xt,yl,56)
          go to 50
40        if(iflag .eq. 1  .and.  (yl-yq)*(yq-yu) .ge. 0.)  yu = yq
          if( int  .eq. 1  .and.  (yl-y2)*(y2-yu) .ge. 0.)  yu = y2
          inout(ii)  =  0
      call drawa(xt,yu)
50        dyl       =  yl - yy
          dyu       =  yu - yy
          dx        =  xc(ii) - xx
          zl        =  a + dx*(b + d*dyl) + c*dyl
```

```
          zu       =  a + dx*(b + d*dyu) + c*dyu
          yzarea(iray)  =  yzarea(iray) + .5*(yu - yl)*(zu + zl)
70        yyp      =  yr(jj)
     if(inout(ii) .eq. 1)  call drawa(xt,yyp)
     if(inout(ii) .eq. 0)  call dasha(xt,yyp,56)
80        continue
100       continue
150       continue
          vol   =  yzarea(1) - yzarea(iray)
          do 200 i=2,iray,2
200       vol   =  vol + 4.*yzarea(i) + 2.*yzarea(i+1)
          vol   =  d4*vol/3.
          return
          end
```

```
      subroutine volume(radrel,vol)
c
c          This subroutine was written and programmed by
c                         A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /index/ nx1,          nx2,          ny1,          ny2,
                     nxgm,         nygm,         xorig,        yorig
      common /volnold/ yzarea
      common /surfac/     xob(1500),    yob(1500),
           zt(1500),      zob(1500),    residu(1500),
           m,             n,            np,
           perd,          resum,        ntc,
           npc,           msq,          msq2,
           msq3,          msq4,         sig
      common /graphc/     xmin,         ymin,         zmin,
           zgrid(65000),  xmax,         ymax,         zmax,
           istat(65000),  nx,           ny,           nz,
           x(16000),      nxgd,         nygd,         ncell,
           y(16000),      delgrd,       igrid,        icell,
           into,          zint,         zlevel,       level,
           noold,         lplot,        dashl,        pcrang,
           levmin,        levmax,       ifit,         isw1,
           isw2,          isw3,         isw4,         isw5
      common /bound/ xb(20000,2),  yb(20000,2),  ncn(2000,2),
                     noch(2),      kod(2000,2),  ipar(2),
                     icnd(2),      kap(15,2),    nptot(2)
      dimension           xc(4),        yr(4),        inout(4)
      dimension           nc(4)
      double precision    yzarea(2001), vol,          zl,
           dx,            dyl,          dyu,          zu,
           a,             b,            c,            d,
           z1,            z2,           z3,           z4
      iflag     =  0
      inreg     =  1
      do 5 i=1,4
5     nc(i)     =  0
      iray      =  4*nx - 3
      do  10 i=1,iray
10    yzarea(i) =  0.
      d4        =  .25*delgrd
      do 150 i=nx1,nx2
      qi        =  i - nx1 - 1
      xx        =  qi*delgrd + xorig
      xc(1)     =  xx + d4
      xc(2)     =  xc(1) + d4
      xc(3)     =  xc(2) + d4
      xc(4)     =  xc(3) + d4
```

```
      call cirobs(xc(1),xc(4),radrel,$150)
      if(kap(15,1) .ne. "no") call bndpnt(xc,nc,ymin,ymax)
      ncsum   =  0
      do 20 l=1,4
      ncsum   =  ncsum + nc(l)
20    inout(l)  =  0
      if(ncsum .eq. 0 .and. kap(15,1) .ne. "no")  go to 150
      do 100 j=ny1,ny2
      qj      =  j - ny1
      yy      =  qj * delgrd + yorig
      yr(1)   =  yy + d4
      yr(2)   =  yr(1) + d4
      yr(3)   =  yr(2) + d4
      yr(4)   =  yr(3) + d4
      call cirobs$ciry(yy,yr(4),radrel,$100)
      kgrid   =  (i - 2)*ny + j
      kgdny   =  kgrid + ny
      if(kgrid .le. 0) kgrid = j
      z1      =  zgrid(kgrid)
      z2      =  zgrid(kgrid+1)
      z3      =  zgrid(kgdny+1)
      z4      =  zgrid(kgdny)
      if(abs(z1) + abs(z2) + abs(z3) + abs(z4)  .le. 0.) go to 100
      a       =  z1
      b       =  (z4 - z1)/delgrd
      c       =  (z2 - z1)/delgrd
      d       =  (z1 - z2 + z3 - z4)/delgrd**2
      do  80 ii=1,4
      if(nc(ii) .eq. 0 .and. kap(15,1) .ne. "no")  go to 80
      if(i .eq. nx1  .and.  xc(ii) .lt. xorig) go to 80
      iray    =  (i - 2)*4 + ii + 1
      yyp     =  yy
      xt      =  xc(ii)
c     call movea(xt,yy)
      do 70 jj=1,4
      yl      =  yyp
      yu      =  yr(jj)
      call cirobs$cirtst(xt,yu,dmin,xxmin,yymin,radrel,$70)
      if(kap(15,1) .ne. "no") call bndpnt$inpnt(yl,yu,yq,nc,ii,inreg,iflag)
      if(inreg .eq. 0  .and.  inout(ii) .eq. 0)  go to 70
      yt      =  yr(jj)
      if(inreg .eq. 0)  yt = yyp
      if(dmin .gt. radrel  .and.  inout(ii) .eq. 0)  go to 70
      if(dmin .le. radrel  .and.  inout(ii) .eq. 1  .and.
         inreg .eq. 1)  go to 50
      call cirint$vlcir(xt,yl,xt,yu,xxmin,yymin,
                        radrel,x1,y1,x2,y2,int)
      if(y1 .lt. yl)  y1 = yl
      if(y2 .gt. yu)  y2 = yu
      if(inout(ii) .eq. 1)  go to 40
      if(iflag .eq. 1  .and.  (yl-yq)*(yq-yu) .ge. 0.)  yl = yq
      if( int  .eq. 1  .and.  (yl-y1)*(y1-yu) .ge. 0.)  yl = y1
      inout(ii)  =  1
c     call dasha(xt,yl,56)
      go to 50
40    if(iflag .eq. 1  .and.  (yl-yq)*(yq-yu) .ge. 0.)  yu = yq
      if( int  .eq. 1  .and.  (yl-y2)*(y2-yu) .ge. 0.)  yu = y2
      inout(ii)  =  0
c     call drawa(xt,yu)
50    dyl     =  yl - yy
```

```
          dyu      =  yu - yy
          dx       =  xc(ii) - xx
          zl       =  a + dx*(b + d*dyl) + c*dyl
          zu       =  a + dx*(b + d*dyu) + c*dyu
          yzarea(iray)  =  yzarea(iray) + .5*(yu - yl)*(zu + zl)
70        yyp      =  yr(jj)
c     if(inout(ii) .eq. 1)  call drawa(xt,yyp)
c     if(inout(ii) .eq. 0)  call dasha(xt,yyp,56)
80        continue
100       continue
150       continue
          vol   =  yzarea(1) - yzarea(iray)
          do 200 i=2,iray,2
200       vol   =  vol + 4.*yzarea(i) + 2.*yzarea(i+1)
          vol   =  d4*vol/3.
          return
          end
```

```
      subroutine window
c
c          This subroutine was written and programmed by
c                         A. C. Olson
c          for use in the GARNET interactive graphics system.
c          GARNET was developed to perform resource mapping
c          and resource estimations for the NATIONAL COAL
c          RESOURCES DATA SYSTEM.
c
c
      common /index/ nx1,          nx2,          ny1,          ny2,
                     nxgm,         nygm,         xorig,        yorig
      common /corner/     xcorn(4),     ycorn(4)
      common /graphc/     xmin,         ymin,         zmin,
           zgrid(65000),  xmax,         ymax,         zmax,
           istat(65000),  nx,           ny,           nz,
           x(16000),      nxgd,         nygd,         ncell,
           y(16000),      delgrd,       igrid,        icell,
           into,          zint,         zlevel,       level,
           nbold,         iplot,        dashl,        pcrang,
           levmin,        levmax,       ifit,         isw1,
           isw2,          ktour,        ismth,        icnt
      common /switch/     isobs,        icrt,         iwind,
           itek,          is5,          is6,          is7,
           is8,           is9,          is10
      dimension           xx(2),        yy(2),        c(69)
1001  format(a4)
1002  format(v)
      write(6,1001)
50    call prompt("CURSOR INPUT? (yes or no): ",27)
      read(5,1001)  itest
      if(itest .ne. "yes"  .and.  itest .ne. "no")  go to 50
      if(itest .eq. "yes")  go to 100
      call bell
      call anmode
      write(6,1001)
      call prompt("XMIN: ",6)
      read(5,1002) xmn
      call bell
      call anmode
      write(6,1001)
      call prompt("XMAX: ",6)
      read(5,1002) xmx
      call bell
      call anmode
      write(6,1001)
      call prompt("YMIN: ",6)
      read(5,1002) ymn
      call bell
      call anmode
      write(6,1001)
      call prompt("YMAX: ",6)
```

```
      read(5,1002) ymx
      go to 200
100   call bell
      is        =  2
      write(6,1001)
      call chnin(is,ierr)
      write(6,1001)
      call obsin(c)
      write(6,1001)
      if(delgrd .le. 0.) call gridin
      call window$unwind
      call anmode
      call crtbeg(xmin,xmax,ymin,ymax)
      if(ierr .ne. 0)  call pltchn(is)
      if(isobs .ne. 0)  call points
      call anmode
      call stty("-modes","^erkl,ctl_char")
110   call bell
      call anmode
      call vcursr(itest,xx(1),yy(1))
      if(itest .ne. 120)  go to 110
      call cross(xx(1),yy(1))
120   call bell
      call anmode
      call vcursr(itest,xx(2),yy(2))
      if(itest .ne. 120)  go to 120
      call stty("-modes","erkl,^ctl_char")
      xmn       =  amin1(xx(1),xx(2))
      xmx       =  amax1(xx(1),xx(2))
      ymn       =  amin1(yy(1),yy(2))
      ymx    _  =  amax1(yy(1),yy(2))
      if(xmn .lt. xmin)  xmn = xmin
      if(xmx .gt. xmax)  xmx = xmax
      if(ymn .lt. ymin)  ymn = ymin
      if(ymx .gt. ymax)  ymx = ymax
      call czaxis(0)
      call movea(xmn,ymn)
      call drawa(xmx,ymn)
      call drawa(xmx,ymx)
      call drawa(xmn,ymx)
      call drawa(xmn,ymn)
      call bell
      call anmode
      read(5,1001)  itest
200   xmnsv     =  xmin
      xmxsv     =  xmax
      ymnsv     =  ymin
      ymxsv     =  ymax
      nx1       =  (xmn - xmin)/delgrd + 1.
      nx2       =  (xmx - xmin)/delgrd + 2.
      ny1       =  (ymn - ymin)/delgrd + 1.
      ny2       =  (ymx - ymin)/delgrd + 2.
      if(nx1 .lt. 1) nx1  =  1
      if(nx2 .gt. nx) nx2  =  nx
      if(ny1 .lt. 1) ny1  = 1
      if(ny2 .gt. ny)  ny2  = ny
      nxgm      =  nx2 - 1
      nygm      =  ny2 - 1
      dm        =  nx1 - 1
      xorig     =  dm*delgrd + xmnsv
```

```
dm          =  ny1 - 1
yorig       =  dm*delgrd + ymnsv
xmin        =  xmn
xmax        =  xmx
ymin        =  ymn
ymax        =  ymx
iwind       =  1
return
entry windsv
xmin        =  xmn
xmax        =  xmx
ymin        =  ymn
ymax        =  ymx
return
entry unwind
xmin        =  amin1(xcorn(1),xcorn(2),xcorn(3),xcorn(4))
xmax        =  amax1(xcorn(1),xcorn(2),xcorn(3),xcorn(4))
ymin        =  amin1(ycorn(1),ycorn(2),ycorn(3),ycorn(4))
ymax        =  amax1(ycorn(1),ycorn(2),ycorn(3),ycorn(4))
dxrng       =  pcrang*(xmax - xmin)
dyrng       =  pcrang*(ymax - ymin)
xmin        =  xmin - dxrng
xmax        =  xmax + dxrng
ymin        =  ymin - dyrng
ymax        =  ymax + dyrng
nx1         =  1
nx2         =  nx
ny1         =  1
ny2         =  ny
nxgm        =  nx - 1
nygm        =  ny - 1
nxgd        =  nxgm
nygd        =  nygm
ncell       =  nxgd*nygd
xorig       =  xmin
yorig       =  ymin
iwind       =  0
return
end
```

```
prompt: proc(prompter,l);
dcl l fixed bin(35), prompter char(64);
put  edit(prompter) (a(l)) skip (0) ;
return; end prompt;
```

This procedure permits the logical (intersection, union, and relative difference) combination of bounded regions. The available operations are:

register -- permits the user to change the principle region to which the corner points of the boundary file must register.

gridset -- reads in a gridded data file for the definition of bounded regions by selection of specified contour levels.

cursor -- permits the definition of a bounded region by use of the graphics display cursor.

combine -- permits the combination of boundaries between two different sets of bounded regions.

display -- displays the boundaries or resultant boundaries from any given boundary file.

quit -- terminates the bounded region combination procedure.

The z-level boundary will be generated from the gridded data and displayed. This boundary may consist of a number of curves, some of which are closed and some of which are open.

Each of these curves will now be displayed in the order in which it was generated. If it is an open curve, you must use the cursor to make it into a closed curve. Normally this will be done by placing the cursor on the intersection of the curve with one of the dashed line quadrangle boundaries and entering the letter "x" from the terminal. Next, you will find the other intersection point of the curve with the quadrangle boundary and again enter a "x".

Continuing in the same manner, you will enter, in order, the desired corner points of the quadrangle. As each point is entered, a small circle will be drawn about the point location. After the last point has been entered, move the cursor back to the first intersection point and enter the letter "t".

After the curve has become closed, the cursor will appear once more. Move the cursor to the side of the closed curve which contains the region that is to be included in further calculations and enter the letter "i", for include, from the keyboard.

If it is desired to skip one of the boundary curves when generating a boundary set, enter the letter "n" the first time the cursor is displayed with that particular boundary component.

To combine different sets of polygonal boundaries:

1) specify the file name for the first set of polygonal boundaries ... these boundaries will be displayed as a set of solid curves.

2) specify the file name for the second set of polygonal boundaries ... these boundaries will be displayed as a set of dashed curves.

Then the cursor will be displayed.

A combination boundary may be identified by first locating an intersection point between the two boundaries of interest and pressing the "x" key. A circle will be drawn about the intersection. Then continue along the boundary segment of interest, locating it with the cursor and pressing the "s" key if the boundary is solid, or the "d" key if the boundary is dashed. If the boundary has been identified, a cross will be drawn at that location.

Next, the subsequent intersection must be located by means of the cursor and by pressing the "x" key. If properly identified, the boundary segment will be retraced with a bold line and the intersection will be identified by means of a circle. Continue in this manner until returning to the starting intersection. Again locate it with the cursor and this time press the "t" key to terminate the boundary closure. A circle with a cross through it will identify the final intersection.

A single closed boundary may be entered into the boundary set by identifying the curve with the cursor and pressing the "c" key. The boundary will be retraced with a bold curve.

After the desired boundary has been selected, the region of interest (inside of or outside of the polygon) must be identified by means of the cursor location followed by entry of the "i" key.

If at any time, an intersection or boundary segment is not properly identified by a circle or a cross, this means that the cursor was not within a close enough tolerance to the curve and the step must be repeated when the cursor again appears.

When all of the desired boundary combinations have been entered, press the "q" key for the next cursor response. The boundary combination procedure will be terminated and a new set of boundary plygons will have been loaded into the boundary array.

The cursor commands are as follows:

x -- identify an intersection point.

```
s    -- tag a branch of the solid boundary.
d    -- tag a branch of the dashed boundary.
t    -- terminate a closed boundary.
c    -- identify a closed curve.
i    -- include region (inside or outside) of polygon.
q    -- end the boundary combination procedure.
```

****

This is GARNET.

It is designed to accept a file of irregularly-spaced data points consisting of z-values associated with given x-, y-coordinate pairs and to process the data in this file to:

1) Generate a fitted surface which describes the behavior of the data.

2) Generate a gridded matrix of z-values.

3) Compute and plot a contour display of this surface.

4) Display those data values which deviate significantly from the general data trend.

5) Permit corrections to be made to the data file for displayed values which are determined to be in error.

6) Perform addition, subtraction, multiplication, or division between the corresponding grid values of two gridded surfaces.

7) Create bounded regions from the gridded data by specifying the desired z-level.

8) Permit the logical combination (union, intersection, and relative difference) between two different boundary sets.

9) Use the gridded data to compute surface-to-surface volumes, tonnages, and to produce plotted resource mapss.

10) Produce suitable hardcopy graphic output for analysis by the geologist.

****

The available operational commands are:

trend: Computes a trend surface fit to irregularly-spaced data. The trend surface may be used to create a gridded data file and for flagging the most deviant data values for examination and verification in the editing procedure.

edit: Displays the trend contour surface and those point values which deviate most extremely from the trend. These extreme values may then be located by means of the cursor and the necessary corrections may be made.

grid: Computes a gridded surface with greater fidelity to the irregularly-located data. The computed grid values may be weighted by the trend surface grid values.

contour: Produces a contour map from a gridded data file.

gridop: Performs addition, subtraction, multiplication, or division between corresponding grid values of two gridded surfaces.

resource: Computes resource volumes and tonnages and plots a resource map for given reliability category distances.

location: Plots the observation point locations with their identifying values.

boundary: Permits the creation of bounded regions from gridded data sets. Also permits the logical combinations of union, intersection, and relative difference, between pairs of boundary sets.

unitconv: This command may be entered whenever a new measurement base, either input or output, and the appropriate scaling factors need to be set. If this command is not used, all scales, horizontal and vertical, and the resulting volumes are presumed to be on a one-to-one basis.